Departamento de Informática

PROYECTO FIN DE CARRERA

CONTROL DE ACCESO EN REDES SOCIALES: INTEROPERABILIDAD Y MINIMIZACIÓN DE LOS DATOS EXPUESTOS

Autor: ALBERTO MARÍN GARCÍA

Tutor: LORENA GONZÁLEZ MANZANO
ANA ISABEL GONZÁLEZ-TABLAS FERRERES

Leganés, septiembre de 2012

Título: CONTROL DE ACCESO GESTIONADO POR EL USUARIO PARA REDES SOCIALES: INTEROPERABILIDAD Y MINIMIZACIÓN DE LOS DATOS EXPUESTOS

Autor: ALBERTO MARÍN GARCÍA

Director: LORENA GONZÁLEZ MANZANO
ANA ISABEL GONZÁLEZ-TABLAS FERRERES

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar a mis padres, mi hermano y mi novia, por el apoyo que me han dado durante toda la carrera y permitir que pudiera dedicarme completamente a ella.

Agradecer también a todos los compañeros que he tenido durante la titulación, ya que cada uno de ellos me ha aportado algo distinto, y todos en común una ayuda muy importante para poder sacar tantos trabajos adelante.

Y a Lorena, mi tutora, por toda la ayuda y los ánimos que me ha dado para que este proyecto pudiera llegar a su fin.

Resumen

En la actualidad nos hemos visto sumergidos en un mundo plagado de tecnología y lo que ella implica, un constante cambio buscando la evolución y la innovación.

Este cambio es destacable en el ámbito de las comunicaciones. Por ejemplo, lo que hace unos años era una carta enviada mediante correo postal, hoy en día es un correo electrónico. Asimismo, las relaciones personales se han visto sustituidas, en gran medida, por el uso de redes sociales.

El desarrollo y uso de las redes sociales, plataformas de intercambio de información entre usuarios y establecimiento de relaciones entre ellos, han superado los límites esperados, conquistando la red informática mundial, conocida como *World Wide Web*. Hay varias muestras de este acontecimiento: dos de estas redes sociales se encuentran actualmente entre las diez páginas más visitadas del mundo (1), más de una decena de estas plataformas posee una cantidad de usuarios en activo superior al centenar de millones (2), las cifras de ingresos que manejan son muy elevadas(2)…

El impacto de esta tecnología nos hace pensar en un mundo absolutamente interconectado, ya no solo a nivel de red tal y como nos lo ofrece internet, sino a un nivel más personal, en el que los usuarios pueden conocerse e interactuar de una manera más sencilla.

Para lograr que esta interconexión sea posible y para mejorar la experiencia del usuario que dispone de varias cuentas en distintas redes sociales, surge el desarrollo de este proyecto. El desarrollo a realizar se basa en crear una plataforma (mediante la implementación de un protocolo), a través de la cual se pueda interoperar entre varias redes sociales. En otras palabras, se pretende ofrecer la posibilidad a los usuarios de poder manejar la información que deseen, compartiéndola en dos o más redes sociales.

El otro aspecto principal que se trata en este proyecto es el de la seguridad de la información personal que los usuarios intercambian con las redes sociales, y que éstas almacenan en sus bases de datos.

Existen diversas leyes relacionadas con la protección de los datos personales, las cuales han de ser cumplidas por cualquier sistema que maneje este tipo de datos y, por tanto, las redes sociales también han de cumplirlas. Ejemplos de estas leyes son la "Ley General de Telecomunicaciones" (Artículo 18.4 de la Constitución Española) o la "Ley Orgánica de Protección de Datos" (3). Aun así, los datos personales pueden verse comprometidos bien por terceras personas que consiguiesen captarlos en el camino existente entre el ordenador del usuario y el servidor, o desde los propios servidores accediendo a las bases de datos, o bien por la falta de ética de alguna de estas empresas que incumplen dichas leyes en pos de mejorar su negocio.

En síntesis, el desarrollo de este proyecto está destinado a cubrir dos necesidades de vital importancia en las redes sociales: la interoperabilidad entre distintas redes sociales para mejorar la experiencia del usuario, y la protección de la información de los usuarios para garantizar su seguridad y mejorar su confianza en estas tecnologías.

Abstract

Currently, we have been inmersed in a world full of technology which leads to the persistent search of improvements and innovation.

This change is remarkable in the field of communication. For example, few years ago we sent letters by tradicional procedures and nowadays, in the majority of cases, we send emails. Furthermore, in multiple cases social relationships have been replaced by social networks relationships.

The development and use of social networks, platforms developed to exchange information and to establish relationships between users, have exceeded the expectations, conquering the World Wide Web. Lots of evidences bear this issue: two of these social networks are among the most visited sites of the world (1), more than a dozen of these platforms have more than one hundred million of active users (2), social networks incomes are very high (2)…

The impact of this technology leads us to think of a completely interconected world, not only at network level, but to a more personal level in which users can easily meet and interact between each other.

To make possible such interconnection and to improve the user experience when he has multiple accounts on different social networks, this project is developed. The development tries to create a platform (implementing a protocol) to interoperate across multiple social networks. In others words, it tries to provide users with tools to carry out the management of their information, considering that their information can be shared among two or more social networks.

On the other hand, this project is also focused on the importance of information security, specially in the exchanged of personal information between users and the secure storage of information in social networks databases.

There are many laws related to personal data protection which must be obey by any system that manages this type of information. Examples of these laws are the “Ley General de las Telecomunicaciones (Artículo 18.4 of Constitución Española)” or “Ley Orgánica de Protección de datos” (3). However, personal data can be compromised either by hackers or third parties entities such as companies that violate these laws towards the improvement of their business.

In conclusion, the development of this project is focused on proposing a solution to the following social networks problems: interoperability between differents social networks to enhance users experiences, and the protection of users information to ensure their information security and improve their confidence in these technologies.

Índice general

Índice de ilustraciones

Índice de tablas

Capítulo 1

– Introducción y objetivos –

1.1 Introducción

Desde que surgiera el concepto de ordenador personal como un computador que aceptaba usuarios de pocos conocimientos informáticos, y se expandiera de forma masiva a lo largo y ancho de todo el planeta, nos hemos acostumbrado a observar como cada cierto período de tiempo, una nueva tecnología modificaba nuestra forma de vida.

Este fenómeno aún se ha visto incrementado en el momento en el que todos estos ordenadores se han podido interconectar, mediante la red de redes por todos conocida, internet.

Por un lado, se ha experimentado un gran cambio en el ámbito laboral, de manera que actualmente un gran porcentaje de la población activa realiza su trabajo haciendo uso de un ordenador, e incluso sin la necesidad de presentarse físicamente en su puesto de trabajo.

Por otro lado, se ha producido una gran variación en las formas de comunicación. En primer lugar, debido a la creación del correo electrónico, el cual sustituyó en gran medida al correo tradicional. Tras el correo electrónico, el intercambio de mensajería instantánea, también conocida como "chat", supuso un nuevo avance en las comunicaciones dado que posibilitó el intercambio de mensajes en tiempo real.

En la última década, la aparición de un nuevo concepto, el de red social, ha dado una nueva vuelta de tuerca a las comunicaciones, convirtiéndose en el modo de comunicación más popular del momento.

Una red social, aplicando el término al campo de la informática, se trata de una plataforma a la que todos los usuarios de internet tienen acceso y, mediante la que es posible conocer e interactuar con gran multitud de personas a nivel mundial. Esto es posible gracias a que las redes sociales son sistemas que posibilitan la creación de un perfil para cada usuario, junto con su posterior vinculación con otros.

Además, las redes sociales ofrecen la posibilidad de, una vez dos cuentas han sido enlazadas, permitir el cambio de todo tipo de información entre ellas, ya sea escrita, mediante imágenes o fotos, o incluso vídeos (teniendo en cuenta que no todas las redes sociales ofrecen los mismos servicios).

Estas posibilidades que ofrecen las redes sociales han suscitado interés entre los usuarios de internet, de modo que la gran mayoría de estos mantiene una cuenta o perfil en alguna red social. Por este motivo, el éxito de estas plataformas ha provocado la aparición de una gran cantidad de ellas.

Como consecuencia de estos dos últimos datos, nos encontramos con que actualmente existen centenares de redes sociales, con mayor o menor éxito o popularidad, y millones de usuarios para muchas de ellas, destacando el caso de Facebook, red social por excelencia, que actualmente posee una cantidad que ronda los 800 millones de usuarios (2).

Sin embargo, debido a la variedad de redes sociales, los usuarios no se conforman con elegir la red social que más se ajusta a sus gustos y características, sino que en una gran medida, han optado por crear una cuenta en varias redes sociales, haciendo uso de dos, tres, o incluso más, siendo ésta una de las motivaciones de este proyecto.

También hay que hacer mención de los aspectos negativos que encontramos en las redes sociales, de los cuales, se destaca la privacidad de los datos.

En el momento en el que un usuario desea registrarse en una de estas aplicaciones web, se le solicitan ciertos datos de carácter personal, sin los cuales no podrá completar la creación de

un perfil en dicha red. Posteriormente, mientras se hace uso de esta plataforma, el usuario puede compartir información, mucha de ella de carácter personal.

Uno de los principales problemas radica en que es difícil conocer con certeza (aun estableciendo las políticas de privacidad correspondientes) todas las personas a las que les será accesible los datos proporcionados a la red social. Es posible que nadie, excepto los usuarios deseados, acceda a los datos, pero ¿qué ocurre con la red social en cuestión? ¿no dispone de los datos y puede aprovecharse de ellos para múltiples propósitos como, por ejemplo, la publicidad?

Teniendo en cuenta el interés despertado por las redes sociales causante de que los usuarios posean cuentas en muchas de dichas redes, y el problema que conlleva la cesión de gran cantidad de información privada, surge este proyecto. Su desarrollo buscará, por un lado, ofrecer interoperabilidad entre redes sociales, y, por otro lado, proporcionar la protección necesaria a los datos de carácter personal para que no sean utilizados deliberadamente por terceros o por las propias redes sociales.

1.2 Estructura del documento

En el presente documento se procederá a describir el proyecto realizado, desde su planteamiento y cálculo de presupuesto, hasta la realización de las pruebas que determinen que el software desarrollado cumple los objetivos establecidos.

El proyecto se estructura en los siguientes capítulos, donde cada uno de ellos trata una fase del proceso de desarrollo o un aspecto importante, tal y como se describe a continuación:

- **Capítulo 1 – Introducción y objetivos:** en este capítulo se realiza una pequeña introducción al proyecto, describiendo a continuación los objetivos que se persiguen. Junto a esto, se expondrá un estudio sobre la situación actual de las redes sociales con mayor número de usuarios.
- **Capítulo 2 – Análisis:** en el segundo capítulo se encuentra un análisis completo del proyecto a realizar. Se comenzará por una descripción de las herramientas que se usarán para su planificación y desarrollo y una especificación a más bajo nivel de los objetivos que se persiguen. Seguidamente, se expondrá la arquitectura del software y sus casos de uso.

  Como parte principal del análisis se presentarán los requisitos del sistema.

  Para finalizar, se expondrá el plan de pruebas, que se deberá llevar a cabo una vez se haya finalizado el desarrollo.
- **Capítulo 3 – Diseño:** sección en la que se encuentra un diseño detallado de la aplicación, sus clases, funciones y métodos. Además, se presentarán los diagramas de secuencia que explicarán el funcionamiento concreto de la herramienta.
- **Capítulo 4 – Implementación y pruebas:** el cuarto capítulo contiene los detalles o problemas tratados durante la implementación de la aplicación cuyo conocimiento se considere importante. Por otro lado, incluye la tabla de revisión de pruebas, en la que se verifica el resultado de cada una de ellas.
- **Capítulo 5 – Simulaciones y medidas:** una vez finalizada la implementación y realizadas las pruebas que verifican el correcto funcionamiento del sistema, se realiza un estudio de los tiempos necesarios para llevar a cabo las acciones más importantes de la aplicación.
- **Capítulo 6 – Conclusión y líneas futuras:** contiene una exposición sobre las conclusiones más importantes extraídas del desarrollo del proyecto, unido a las posibles mejoras que se podría realizar sobre éste.
- **Capítulo 7 – Referencias:** durante la lectura del documento se encuentran en diversas ocasiones referencias a sitios web o documentación. Estas referencias se indicarán siguiendo la Norma ISO 690-2 (4) para documentos electrónicos. En este capítulo se encontrarán todas estas referencias ordenadas en función de su orden de aparición en el documento.
- **Capítulo 8 – Glosario de términos:** esta sección agrupará todas las definiciones de los términos que puedan llevar a confusión, así como de los acrónimos que aparezcan en el documento.

1.3 Objetivos del proyecto

Tal y como se mencionó en la introducción, el proyecto a desarrollar presenta dos objetivos principales: proporcionar al usuario la posibilidad de interoperar con varias redes sociales y, dotar a sus comunicaciones de las técnicas de seguridad necesarias para que los riesgos causados por la exposición de sus datos personales a terceros se minimicen en la medida de lo posible.

A continuación, se describen cada uno de estos objetivos.

1.3.1 Interoperabilidad entre redes sociales

El término "interoperabilidad" se refiere al intercambio o puesta en común de datos entre dos sistemas distintos. Por tanto se puede definir el primer objetivo del proyecto como la posibilidad de compartir datos de perfil y fotografías entre distintas redes sociales.

Para cumplir este objetivo se desarrolla una nueva arquitectura en el entorno de las redes sociales, una arquitectura en la que los datos de los usuarios, así como sus imágenes o publicaciones, se almacenan en servidores dedicados exclusivamente a este propósito, y de los cuales las redes sociales extraerán la información que mostrarán en sus interfaces. Más concretamente, son servidores descentralizados cuyo propósito es el almacenamiento de la información de los usuarios, la cual no pertenecerá a ninguna red social concreta, consiguiendo que varias de estas redes puedan obtener los datos de un mismo servidor y así, lograr la interoperabilidad.

Existirán dos tipos de servidores encargados de almacenar la información privada del usuario: los encargados de almacenar los datos referentes al perfil (nombre, email, lista de contactos…) y los encargados de guardar los recursos del usuario (fotografías). De esta forma el usuario almacenará sus datos en el servidor correspondiente, y tan solo tendrá que proporcionar a la red social en la que desee iniciar sesión, las direcciones de estos servidores y su dirección personal de correo. Este paso se realizaría en cada una de las redes sociales en las que se desee iniciar sesión, de manera que todas podrían hacer uso de estos datos (siempre y cuando el usuario lo desee).

Para obtener aún un nivel mayor de interoperabilidad, y tal y como se detallará en el análisis, será posible obtener los datos que uno de los contactos usa en una red social distinta.

Cabe mencionar, a pesar de que a lo largo del documento se profundizará más en este aspecto, que en todo momento el usuario propietario de los datos será quien decida qué redes sociales o contactos pueden acceder a sus datos y recursos.

1.3.2 Minimización de los datos expuestos

El segundo objetivo del proyecto se basa en la minimización de los riesgos a los que se enfrentan los datos de los usuarios de las redes sociales. Se entiende por minimización de los datos expuestos el hecho de que las redes sociales no tengan conocimiento de los datos privados del usuario. De esta forma, se evitaría la posibilidad de que las grandes redes

hiciesen un mal uso de la información privada de los usuarios, o incluso accediesen a venderla a empresas del sector del marketing (5).

Para la consecución de este objetivo, la información se almacenará cifrada en servidores externos, evitando el acceso de las redes sociales a la información.

El acceso a la información de los usuarios pasa de esta forma a estar custodiado por estos servidores externos, frente a los cuales un usuario debe autenticarse en el momento en que se desee hacer uso de una red social.

Con el fin de simplificar el protocolo, en esta implementación la autenticación se va a realizar a través de la red social, por lo tanto se comentará la necesidad de realizar, como futura mejora, la autenticación frente a los servidores.

Junto al cifrado de la información, se hace uso de otros mecanismos de seguridad como la firma electrónica. Este mecanismo, usado en momentos clave del protocolo (acceso a información y recursos), comprobará que el usuario que intenta recuperar la información es el mismo que realiza el inicio de sesión, y no se trata de un ataque de "Man in the middle".

Respecto a los mensajes intercambiados entre los servidores de almacenamiento de información y el propio servidor de la red social, es importante destacar que siguen el protocolo SSL, para garantizar que no sufren ataques a la confidencialidad ni a la integridad y, asimismo, existe autenticación frente al servidor. Además, a la hora de recuperar las fotografías, el usuario que realiza el acceso proporcionará una clave pública para que se cifren éstas, de manera que solo él pueda descifrarlas en su propio equipo.

1.4 Estado del arte

En esta sección se describe la situación actual en la que hará aparición el proyecto a desarrollar, mostrando sistemas similares y detallando las posibilidades que ofrecen las actuales redes sociales. El estudio de las redes sociales actuales se centrará en el aspecto de la interoperabilidad, debido a que el segundo punto tratado en el proyecto, la minimización de los datos expuestos, hoy día no es contemplado por ninguna de estas redes.

Junto a esta descripción, se resalta la innovación que supone el desarrollo de este proyecto en el ámbito de las redes sociales.

Como se expuso en la introducción de este documento, es un hecho conocido que multitud de usuarios de las redes sociales poseen un perfil en varias plataformas simultáneamente.

Este hecho hace suponer que las empresas propietarias de redes sociales han dotado a éstas de la posibilidad de interoperar entre ellas, con el fin de proporcionar al usuario mayor usabilidad. Sin embargo, existen grandes limitaciones en el ámbito de la interoperabilidad entre redes sociales, tal y como se ha podido observar en el estudio aquí presentado sobre algunas de las redes sociales más importantes a nivel internacional.

A continuación, se realiza una breve descripción de la situación actual de Facebook, Twitter, Badoo, LinkedIn, Orkut, Google+ y MySpace, y posteriormente se expone una tabla resumen con las conclusiones del análisis.

1.4.1 Facebook

Ilustración 1: Facebook

Facebook (6) es, a día de hoy, la red social con mayor cantidad de usuarios registrados, superando la cifra de los ochocientos millones, con la cual casi duplica a la segunda red social con más usuarios, Qzone, con alrededor de cuatrocientos ochenta millones (la cual no se analizará por estar enfocada en el continente asiático, China en concreto), y casi triplica los de la tercera, Twitter, con 300 millones de usuarios.

En cuanto a la interoperabilidad que ofrece Facebook, cabe destacar que no existe la posibilidad de compartir información desde su web hacia afuera, lo cual limita las posibilidades y funcionalidades al alcance de los usuarios.

Cuando se trata de realizar la operación en el sentido contrario, la situación cambia. Facebook ofrece la posibilidad de configurar, a través de aplicaciones, que el resto de redes sociales creen y publiquen en su web, compartiendo de manera sincronizada toda la información que un usuario exponga. Esto mejora notablemente su interoperabilidad siempre y cuando el resto de redes sociales hayan desarrollado estas aplicaciones, lo cual se tratará en posteriores análisis.

1.4.2 Twitter

Ilustración 2: Twitter

Twitter (7) es la segunda red social de carácter internacional más grande del mundo, con alrededor de trescientos millones de usuarios activos, y junto a Facebook compone el dúo de redes sociales que se encuentran en el "top 10" de páginas webs más visitadas del mundo, ocupando la novena posición (2).

Esta red social, basada principalmente en compartir mensajes cortos (no más de 140 caracteres) con otros usuarios, tiene una aplicación a través de la cual compartir información con Facebook. Para ello, el usuario debe activar esta opción en su configuración personal, y autenticarse en Facebook con sus respectivas credenciales. Tras esta configuración, cualquier mensaje que un usuario publique en su tablón de Twitter aparecerá también reflejado en su muro de Facebook.

Pese a que Twitter ofrece esta posibilidad de interoperabilidad, no se considera suficiente, ya que la interoperabilidad es más sencilla y menos costosa cuando la información son mensajes cortos de texto (caso de Twitter), que cuando se trata de imágenes o vídeos.

1.4.3 Badoo

Ilustración 3: Badoo

Se trata de una red social que, actualmente, tal y como muestran en su página principal, posee aproximadamente ciento treinta y siete millones de usuarios en todo el mundo.

Badoo (8) posee una mayor interoperabilidad que las redes analizadas anteriormente, dado que permite al usuario autenticarse en el sistema haciendo uso de los datos de login usados en Facebook ("username" y "password").

Una vez el usuario ha iniciado su sesión e Badoo, con los datos correspondientes a esta red social, o con los de Facebook, encontrará una nueva forma de interoperar con otras redes sociales. En este caso, el usuario puede importar fotografía o imágenes que estén asociadas a su perfil en otras tres redes sociales (Facebook, MySpace y Orkut).

Observando las características anteriores que ofrece Badoo, se podría pensar que la situación en referencia a la interoperabilidad es bastante alentadora. Sin embargo, en esta red social contemplamos un problema grave de seguridad, que se extenderá a todas las redes sociales con las que se haya interoperado.

El problema principal es que Badoo ha sido catalogada como una de las peores redes sociales según un estudio realizado por la Universidad de Cambridge (9), en el cual se contemplaba la privacidad de los datos expuestos en las diversas redes sociales. Por tanto, se puede suponer que, al ser necesario proporcionar los datos de autenticación de toda aquella red social con la que se desease interoperar, estos pueden llegar a terceras personas dada la poca seguridad que ofrece Badoo. Si esto ocurriera, todos los datos personales de los usuarios podrían llegar a estar comprometidos, lo cual resulta inadmisible.

Tras este análisis, es posible indicar que el beneficio que aporta Badoo en cuanto a interoperabilidad se refiere, se ve ampliamente reducido por la falta de seguridad de dicha red social.

1.4.4 LinkedIn

Ilustración 4: LinkedIn

LinkedIn (10) es la red profesional con mayor número de usuarios del mundo, considerada a su vez una red social. Entre sus aproximadamente ciento veinte millones de usuarios destaca la aparición de perfiles de empresas, lo que permite al resto de usuarios establecer contacto con éstas, siendo una nueva forma de encontrar empleo.

La interoperabilidad que ofrece esta red social se reduce a la posibilidad de compartir las actualizaciones realizadas en LinkedIn con Twitter, para lo cual será necesario iniciar una sesión en esta segunda red, y establecer un permiso explícito para LinkedIn. Esta forma ofrece una mayor seguridad que la de Badoo, pero de nuevo encontramos una interoperabilidad muy reducida.

1.4.5 Orkut y Google+

Ilustración 5: Orkut

Ilustración 6: Google+

Orkut (11) y Google+ (12) pertenecen a la compañía Google, y por este motivo, la interoperabilidad ofrecida se orienta a productos desarrollados por dicha compañía.

Ninguna de las dos redes permite interoperar con otra red social que se encuentre fuera del ámbito de Google. Sin embargo, ambas pueden interoperar con otros productos pertenecientes a la compañía.

Por un lado, estas plataformas ofrecen la posibilidad de autenticación haciendo uso de los datos de login de la cuenta de correo electrónico de Gmail con la que se haya realizado el registro.

Por otro lado, ofrecen la posibilidad de buscar contactos que posean un perfil en cada una de las redes sociales a través de la agenda de contactos que se almacena en el servidor de correo electrónico.

Cabe destacar que esta última posibilidad la ofrecen la mayoría de las redes sociales, si bien no ha sido aún comentada por no considerar interoperabilidad entre redes sociales, sino entre una red social y un servidor de correo electrónico.

1.4.6 MySpace

Ilustración 7: MySpace

Por último se analiza MySpace, red social de aproximadamente cien millones de usuarios registrados, diseñada para compartir, especialmente, intereses o aficiones a través de música y vídeos.

Es importante mencionar esta red social ya que permite la sincronización tanto con Facebook como con Twitter, lo cual la diferencia del resto de redes sociales analizadas.

Para que esta sincronización sea posible, se debe iniciar sesión en cada una de ellas, y otorgar los permisos necesarios. Una vez realizadas estas tareas, cualquier información compartida en MySpace aparecerá en los correspondientes muros de las redes sociales mencionadas.

1.4.7 Conclusiones del análisis

La gran mayoría de las redes sociales existentes en la actualidad ofrecen escasas opciones para permitir interoperabilidad. En general, únicamente es posible interoperar entre una o dos redes sociales, permitiendo exponer en el muro de una red social información publicada en una segunda red.

A parte de esto, algunas redes sociales no poseen suficientes mecanismos de seguridad para garantizar la privacidad de la información, lo cual supone un grave problema para los usuarios.

En la siguiente tabla, se muestra un resumen de las posibilidades que ofrece cada red social actual en los aspectos de interoperabilidad y protección de la información.

Red Social	Interoperabilidad con alguna SN	Interoperabilidad con al menos tres SN	Información asegurada frente a atacantes	Información asegurada frente a la propia SN
Facebook	✓	✗	✓	✗
Twitter	✓	✗	✓	✗
Badoo	✓	✓	✗	✗
LinkedIn	✓	✗	✓	✗
Google+ y Orkut	✗	✗	✓	✗
MySpace	✓	✗	✓	✗

Tabla 1: Conclusiones del estudio del arte

En resumen, el proyecto que se describe a lo largo de este documento aporta la interoperabilidad que, como se observa en la tabla, ninguna de las redes sociales actuales ofrece (excepto en situaciones concretas). Tal es el grado de interoperabilidad ofrecido por esta aplicación que permite desde utilizar un mismo perfil para todas las redes sociales en las que se esté registrado, hasta utilizar uno para cada red social y que los contactos elijan de cual desean obtener la información.

Por otro lado, ofrece la protección de la información frente a agentes externes de la cual algunas de estas redes carece, pero sin duda posee una mayor importancia el hecho de que ésta información se proteja también de las propias redes, evitando problemas como la venta de datos a empresas de marketing.

Para finalizar, destacar que la información que la aplicación va a permitir compartir serán los datos de perfil y fotografías de los usuarios, sin embargo, con una pequeña mejora, se podría adaptar para compartir otra información como comentarios, vídeos, audios, etc.

Capítulo 2

– Análisis –

2.1 Especificación de los objetivos

En esta sección se realizará una especificación de los objetivos descritos en la Sección 1.3 de este documento. Para ello se dividirá la explicación en dos subsecciones diferentes.

La primera de ellas se dedica a la interoperabilidad entre redes sociales, es decir, explica el protocolo que se debe seguir y cómo son los mensajes intercambiados.

La segunda subsección contiene la explicación del segundo de los objetivos, la minimización de los datos expuestos, qué es y cómo se consigue. Además, se van a describir otros aspectos de seguridad que actuarán de manera complementaria en la protección de la información.

2.1.1 Descripción del protocolo

El protocolo que se va a seguir es un protocolo genérico, es decir, cumple los dos objetivos perseguidos en el proyecto, la interoperabilidad y la minimización de datos expuestos.

A continuación se describirán las entidades que participan en el protocolo, así como el tipo y contenido de mensajes que intercambian. No obstante, se ha de considerar que en este protocolo se hace uso de datos de perfil, especificados en ficheros .foaf, y de recursos, concretamente fotografías. Por este motivo, se va a comenzar por detallar qué es un fichero FOAF y cuál es su contenido.

2.1.1.1 Ficheros FOAF

Este tipo de ficheros será utilizado para almacenar y transportar los perfiles de los usuarios, en los que se incluyen los datos personales y las relaciones establecidas con el resto de usuarios, tal y como se detallará a continuación.

Las siglas FOAF, cuyo significado en inglés es *Friend of a friend* (“Amigo de un amigo”), dan nombre al proyecto encargado de proporcionar una forma más fácil de compartir y utilizar información sobre las personas y sus actividades. (13)

Este tipo de ficheros mantiene una estructura en árbol, similar a la de los XML, en la que cada nodo puede contener un par clave-valor, o varios nodos más.

Gracias a librerías como GSON, se puede obtener este formato a partir de un objeto JAVA con tan solo una instrucción. Este proceso es reversible, por lo que también es posible obtener un objeto JAVA a partir de un fichero FOAF de manera sencilla.

Esta cualidad consigue que el trabajo con FOAF sea muy rápido, ya que se no es necesario “parsear” el fichero cada vez que se reciba, sino que se puede transformar en un objeto y trabajar con él de manera más cómoda.

Es importante destacar que los objetos que se pueden serializar no deben ser necesariamente propios del lenguaje JAVA, sino que pueden ser definidos por el propio desarrollador. De esta forma, se determina que los atributos que forman el objeto que representa el perfil y, por tanto, los campos del fichero FOAF, aun pudiendo existir muchos otros, son los siguientes:

- **Name**: nombre del usuario propietario del perfil.
- **Email:** dirección de correo electrónico
- **Nationality:** procedencia del usuario (dado que la aplicación soportará redes de ámbito internacional).
- **WBSN:** lista de redes sociales en las que el usuario está registrado
- **School:** centro de estudios al que pertenece el usuario. Campo clave en la verificación de políticas de acceso
- **Age:** edad del usuario. Campo clave en la verificación de políticas de acceso.
- **Friends:** nodo que se compondrá de dos nodos inferiores:
    - **Friend_email:** dirección de correo electrónico con la que se identificará al contacto
    - **Friend_WBSN:** lista de redes sociales en las que se encuentra registrado el contacto

Conociendo los campos que conforman el fichero FOAF que describe los perfiles de los usuarios, es momento de conocer la estructura final de estos una vez se serializan en RDF:

```
<rdf:RDF>
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

        <foaf:Person>
                <foaf:name>Alberto Marín</foaf:name>
                <foaf:mbox rdf:resource="amg.tfg@gmail.com" />
                <att:nationality>Spanish</att:nationality>
                <att:WBSN>friendbook,myleisure</att:WBSN>
                <att:school>UC3M</att:school>
                <att:age>22</att:age>
                </foaf:Friend>
                        <att:friend_email>a@gmail.com</att:friend_email>
                        <att:friend_wbsn>friendbook</friend_wbsn>
                </foaf:Friend>
        </foaf:Person>
</rdf:RDF>
```

2.1.1.2 Descripción de las entidades participantes

Antes de comenzar la descripción del protocolo de mensajes, se deben exponer cuáles serán las entidades participantes en éste. Estas entidades son:

Entidad	Función en el protocolo
SN (Social Network)	Proporciona la interfaz a través de la cual el usuario interactuará con el sistema. Además, debe proporcionar las conexiones necesarias con que el resto de servidores puedan llevar a cabo la funcionalidad requerida.
IdP (Identity Provider)	Servidor que almacena la información del perfil de los usuarios (contenida en ficheros FOAF que se detallan en la Sección 2.1.1.4), así como sus correspondientes certificados para poder realizar firmas. Será el encargado de recibir las peticiones de información desde las redes sociales.
AM_IdP (Authentication Manager - IdP)	Este servidor es el encargado de almacenar las políticas de seguridad de los usuarios. Determina si la red social y el usuario con los que contacta el IdP son aptos para acceder a la información que desean.
Host	Servidor que almacena los recursos de los usuarios (en este proyecto tan solo se tratan fotografías, pero en posteriores mejoras podrían tratarse comentarios, vídeos…). Es el encargado de recibir las peticiones de recursos desde las redes sociales.
AM_Host (Authentication Manager - Host)	Actúa de igual manera que el AM_IdP, con la diferencia de que determina el acceso a los recursos en lugar de a la información del perfil.

Tabla 2: Entidades participantes en el protocolo

Conociendo las entidades participantes y su función dentro del protocolo, se presenta el siguiente esquema que muestra cómo se interconectan para poder llevar a cabo las comunicaciones necesarias.

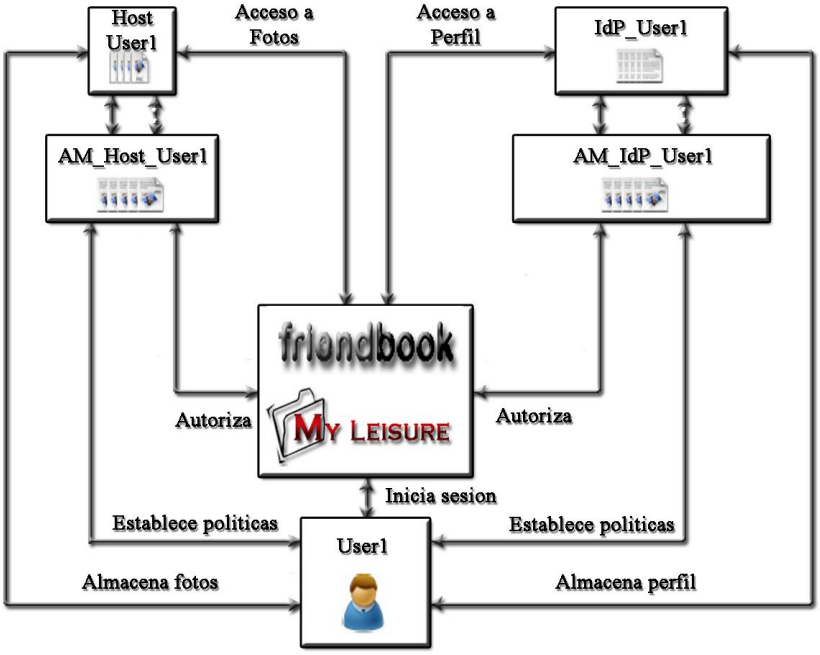

Ilustración 8: Interconexión de entidades

Estos cinco tipos de entidades serán los participantes en el protocolo de intercambio de mensajes que se seguirá en la implementación. De ellas se deben comentar ciertas particularidades para comprender su funcionamiento a la hora de escalar el proyecto y trabajar con más de una red social y más de un usuario.

En primer lugar, la entidad *SN* se verá representada por todas y cada una de las redes sociales existentes en la actualidad que deseasen hacer uso de este protocolo.

Como se comenta en la tabla, los *IdP* y los *Host* son los encargados de guardar los datos y recursos de los usuarios. Por lo tanto, pueden existir desde un único servidor de cada tipo (un *IdP* que almacene los perfiles de todos los usuarios, y un *Host* que haga lo mismo para los recursos), hasta un servidor de cada tipo para cada uno de los usuarios registrados (casuística que se empleará en este proyecto).

Para finalizar, falta indicar que cada *IdP* se encuentra ligado a un *AM_IdP*, que verifica que el usuario que solicita un recurso cumple las políticas de acceso. De la misma manera ocurre con los *Host* y los *AM_Host*.

Estas políticas de acceso son establecidas por los usuarios en el momento del registro, pudiendo ser modificadas posteriormente. Se basan en dos campos: school y age. Los usuarios pueden determinar que tan solo accedan a su información aquellos contactos que pertenezcan a un rango de edad determinado. De igual manera, pueden decidir el centro de estudios al que deben pertenecer (o en caso contrario, al que no deben pertenecer).

Se ha de destacar que en futuras mejoras se podría determinar que existiese un *AM* genérico, que se encargase de realizar las autenticaciones tanto para los *IdP* como para los *Host*.

2.1.1.3 Definición del protocolo de mensajes

En el protocolo desarrollado se diferencian dos partes, la obtención de datos de perfil y recursos (fotografías) cuando un usuario inicia sesión en una red social, y la obtención de datos de perfil y recursos de un contacto de un determinado usuario. En estas partes, se intercambian una cantidad distinta de mensajes y participan distintas entidades. Sin embargo, el formato de los mensajes es muy similar.

En primer lugar se describirá la parte del protocolo cuyo objetivo será el de obtener tanto los datos del perfil, como los recursos de un usuario que inicia sesión en una red social. En los siguientes esquemas se pueden observar los mensajes intercambiados con estos fines.

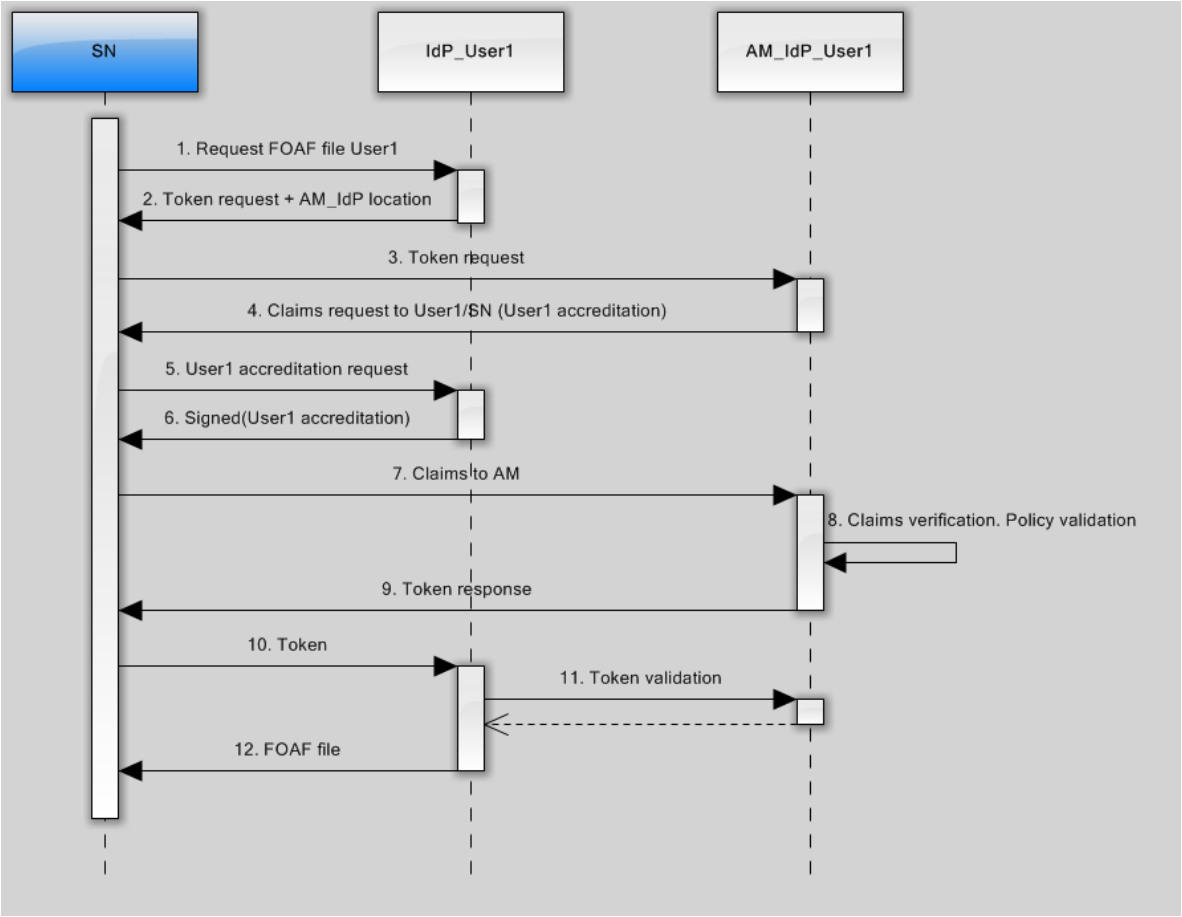

Ilustración 9: Protocolo para la obtención del perfil de un usuario

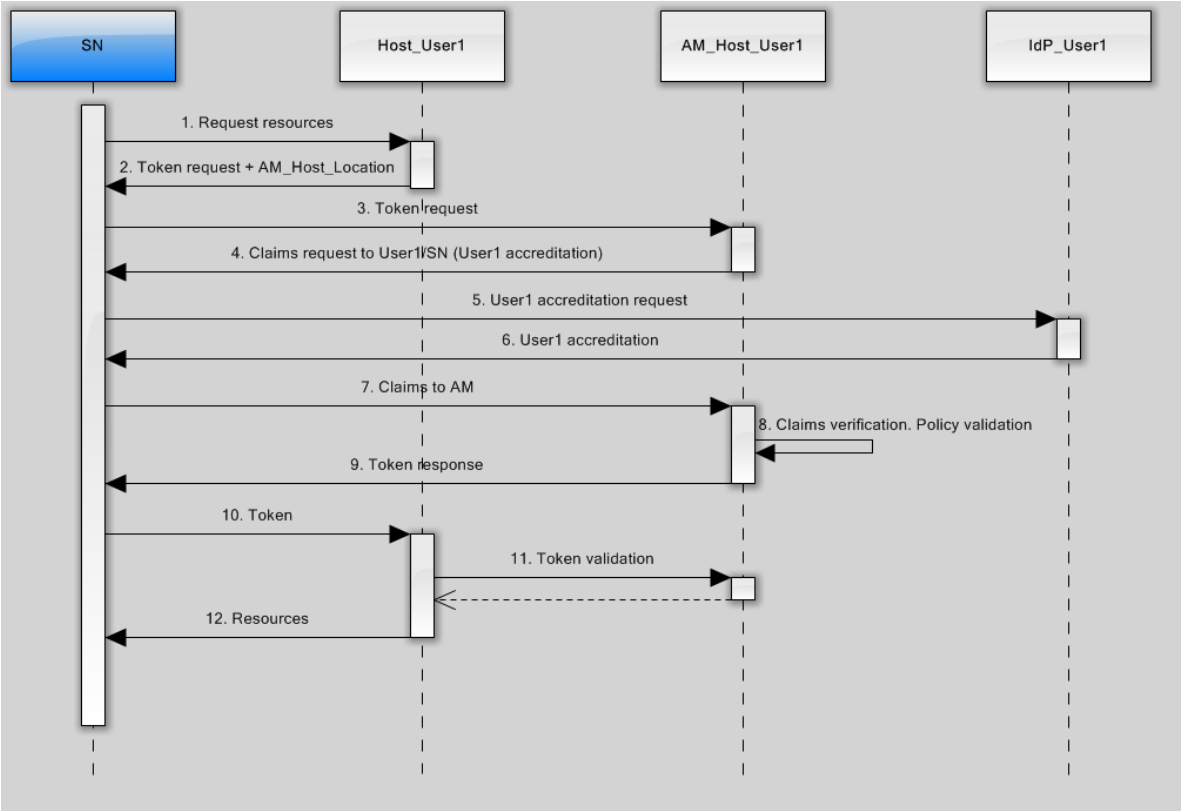

Ilustración 10: Protocolo para la obtención de recursos de un usuario

A continuación se explica el significado de cada uno de los mensajes, teniendo en cuenta que cada mensaje se identifica con un número.

1) La red social solicita al *IdP* que contiene los datos del usuario que ha iniciado sesión, los datos de su perfil. En este momento se debería alertar a los servidores *IdP* y *Host*, de que un usuario requerirá sus datos (esta medida no se implementará, por lo que se mantendrán ambos servidores escuchando todos los mensajes en cualquier momento).
2) El *IdP* contesta a la petición, indicando que para poder facilitar esta información se le debe proporcionar un "token" otorgado por el *AM_IdP*. En este mismo mensaje se incluirá la dirección de esta tercera entidad para que la red social pueda contactar con ella.
3) La red social solicita al *AM_IdP* el "token" requerido por el *IdP*.
4) El *AM_IdP* contesta al mensaje solicitando una acreditación del usuario, es decir, algún mecanismo que indique que el usuario es quien dice ser.
5) La red social contacta de nuevo con el *IdP* para que éste le proporcione la acreditación del usuario.
6) El *IdP* contestará a la petición de la red social, proporcionándole la acreditación solicitada. Esta acreditación constará de un pequeño texto formado por el "hash" del email del usuario concatenado a la fecha actual del sistema. Este texto se firmará (con la clave privada del *IdP*), consiguiendo autenticar al usuario.
7) La red social proporciona al *AM_IdP* la acreditación que éste le solicitó.

8) El *AM_IdP* verifica la firma (con la clave pública del *IdP*). Si todo es correcto, verifica la/las políticas de control de acceso establecidas por el usuario del que se solicita los datos.
9) Si se cumplen las políticas de control de acceso, se proporciona a la red social el "token" para que pueda recibir los datos del perfil del usuario.
10) La red social entrega al *IdP* el "token" recién obtenido, y queda a la espera de recibir los datos del usuario.
11) El *IdP* contacta con el *AM_IdP* con un mensaje en el que le entregará el "token" que le ha entregado el usuario. El objetivo de este mensaje es que el *AM_IdP* compruebe que el "token" recibido por el *IdP* es el mismo que acaba de generar momentos antes, lo cual indicaría que la autenticación ha sido exitosa. De ser así, se procede al envío del mensaje "12".
12) El *IdP* proporciona a la red social los datos del perfil del usuario. Estos se proporcionarán a través de un fichero "FOAF", el cual cumple un formato específico que describirá en próximas subsecciones.

De esta forma, la red social conseguiría los datos del perfil del usuario, para acto seguido mostrarlos en su interfaz.

Como se observa en la Ilustración XX, los mensajes intercambiados con el fin de obtener las fotografías serán los mismos que los anteriormente detallados, salvo que la solicitud se realiza entre *AM_Host* y *Host*.

Una vez descrita la parte del protocolo encargada de obtener los datos del usuario una vez iniciada la sesión, se continuará con la parte más compleja y que, especialmente, se enfoca en satisfacer el objetivo de interoperabilidad. Esta segunda parte se basa en la obtención de datos de un segundo usuario (en adelante *User2*) que posee algún vínculo con el primero (*User1*), es decir, ha establecido una relación, la cual aparecerá en el fichero FOAF que almacena los datos de perfil.

Para conseguirlo, esta fase entrará en funcionamiento una vez el *User1* seleccione a uno de los usuarios de su lista de contactos y, acto seguido, elija la red social de éste segundo de la que desea obtener los datos. En el momento en el que se seleccione esta red social, se ejecutará el siguiente intercambio de mensajes:

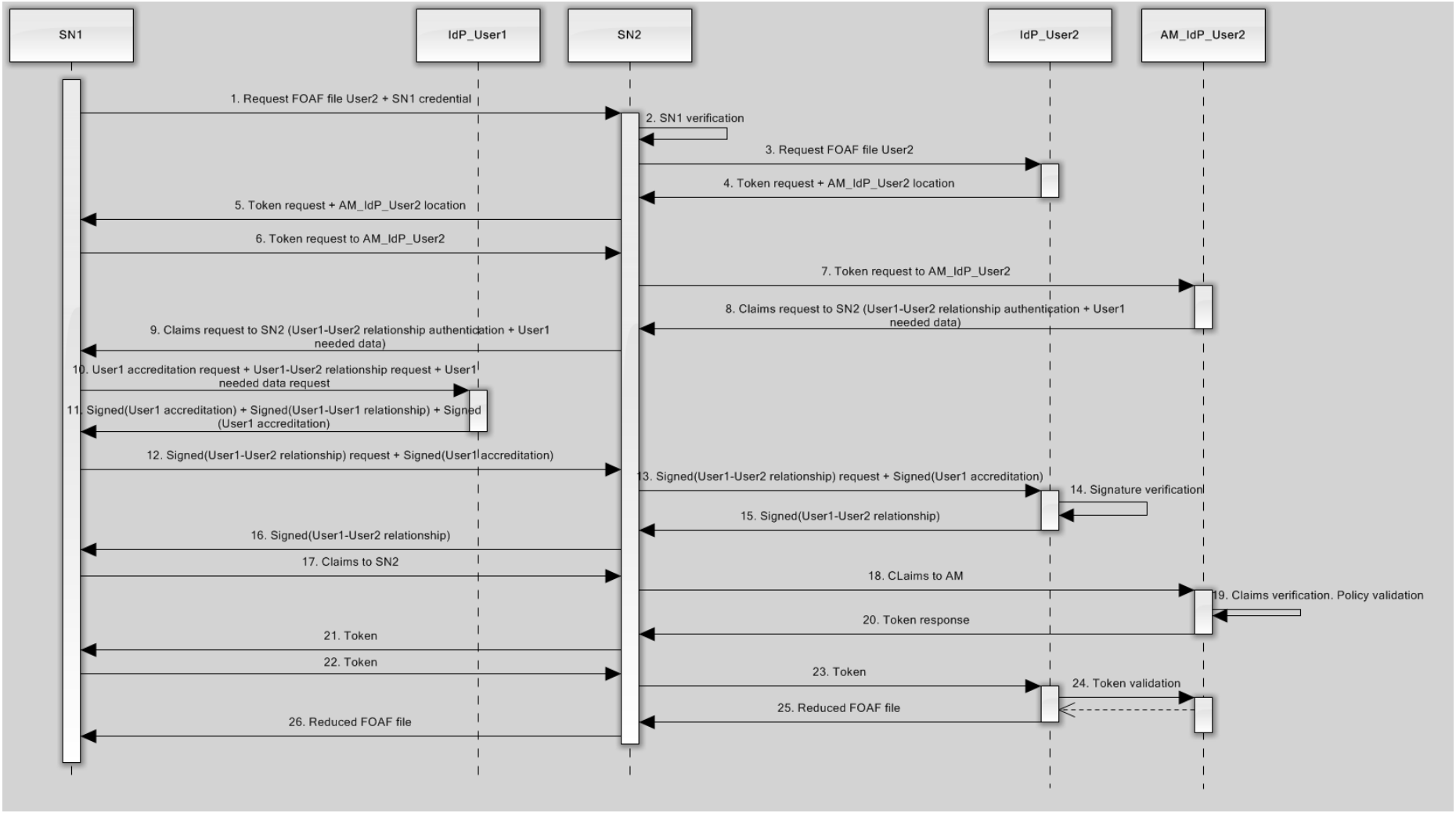

Ilustración 11: Protocolo para la obtención de información de un contacto

De igual manera que sucedió con la primera parte del protocolo, cada mensaje posee un identificador numérico:

1) La primera red social (*SN1*) solicita a la segunda (*SN2*) los datos del perfil de *User2*. En este mensaje, *SN1* incluirá su firma para que *SN2* pueda comprobar que la solicitud proviene de una red social real y no se trata de un ataque.
2) *SN2* verifica que la firma de *SN1* es correcta, lo cual confirma que no se trata de un ataque.
3) La *SN2* solicita al *IdP_User2* (servidor en el que *User2* almacena sus datos) el perfil del segundo usuario.
4) El *IdP_User2* solicita a *SN2* un "token" para poder entregarle la información que solicita. En este mensaje se incluye la dirección del *AM_IdP_User2*, con el cual se deberá establecer contacto para obtener el "token" solicitado.
5) *SN2* comunica a *SN1* que no podrá proporcionarle ninguna información hasta que no se le proporcione un "token" que verifique la autenticación.
6) *SN1* envía un mensaje a *SN2* pidiéndole que establezca comunicación con el *AM_IdP_User2* con el fin de obtener el "token".
7) *SN2* solicita a *AM_IdP_User2* el "token".
8) *AM_IdP_User2* contesta al mensaje solicitando una acreditación que identifique a *User1*, y que se compruebe la relación entre ambos usuarios.
9) *SN2* solicita a *SN1* la entrega de ambas acreditaciones.
10) *SN1* pide al *IdP_User1* la acreditación para este usuario.
11) El proceso de acreditación se realiza tal y como se detalló en la primera parte del protocolo, es decir, mediante la aportación de la firma por parte de *IdP_User1*. Una vez realizada la firma, se manda la acreditación a *SN1*.
12) Se manda la acreditación firmada desde *SN1* hacia *SN2*.
13) *SN2* traslada esta acreditación al *IdP_User2*, para que verifique que es correcta.
14) Se verifica que la firma es correcta.
15) Si todo es correcto, se devuelve esta acreditación a *SN2*.
16) *SN2* traslada de nuevo la acreditación firmada a *SN1*. El motivo de que se devuelva la acreditación a la primera red social es que, en caso de repetirse el proceso de obtención de información de un contacto, este se verá acelerado al mantener la acreditación.
17) *SN1* manda nuevamente la acreditación a *SN2*, esta vez con el objetivo de que la haga llegar a *AM_IdP_User2*, el cual deberá generar el "token".
18) *SN2* traslada la acreditación hacia *AM_IdP_User2*.
19) *AM_IdP_User2* verificará de nuevo la firma de la acreditación para comprobar que no ha habido ningún problema en los anteriores mensajes. Si la firma no se ha visto alterada, se procederá a comprobar las políticas de seguridad del *User2*. En esta comprobación de políticas, lo que se hará es verificar que el *User1* cumple con los requisitos de edad y de procedencia escolar que el *User2* desea. De ser así, se procede con el siguiente mensaje.
20) *AM_IdP_User2* genera un "token" y se lo manda a *SN2*.

21) *SN2* traslada el "token" a *SN1*
22) *SN1* devuelve el token a *SN2*, para que esta se lo haga llegar al *IdP_User2*.
23) *SN2* envía el "token" al *IdP_User2*.
24) El *IdP_User2* contacta con *AM_IdP_User2* para comprobar que el "token" es válido.
25) Una vez realizada con éxito la autenticación, el *IdP_User2* proporciona a *SN2* el fichero "FOAF" que contendrá los datos del perfil de *User2*.
26) Para finalizar el protocolo, "SN2" proporciona los datos solicitados a "SN1".

Comparando esta parte del protocolo con la anterior, se puede comprobar que son similares, teniendo como únicas diferencias los intercambios de mensajes entre ambas redes sociales, y la acreditación de la relación entre ambos usuarios.

Como última parte del protocolo, aparece la dedicada a obtener los recursos de *User2* almacenados en un *Host* distinto al de *User1*. Este fragmento del protocolo no se detallará ya que es igual que el que se acaba de describir, salvo algunas diferencias que solo afectan al cifrado de datos y al contenido de los mensajes, por lo que se comentarán posteriormente.

2.1.1.4 Formato de los mensajes

Los mensajes intercambiados durante el uso de la aplicación, y definidos en el protocolo como anteriormente se detalló, serán mensajes HTTP, bien de tipo POST o de tipo GET.

Todos estos mensajes estarán formados por una parte fija, es decir, unos campos que se podrán encontrar en cualquier mensaje de los que conforman el protocolo. Por otro lado, se observará una parte variable, que dependerá del tipo de mensaje que se analice y del objetivo de éste.

En primer lugar, se describirán los campos fijos, por afectar a todos los mensajes de la aplicación:

Campo	Descripción	Funcionalidad
Type of Message	Se nombrará como tipo de mensaje al identificador de estos: requestFoafFile, requestResources, tokenRequest…	Su funcionalidad es doble: por un lado, se usará para que cuando un servidor recibe un mensaje, sepa cómo actuar ante éste. Por otro lado, será útil para evitar duplicidades de código en los casos en los que dos mensajes pertenecientes a diferentes secciones del protocolo sean similares.
User	Posee un tamaño de 20 Bytes. Su contenido será el hash de la dirección email proporcionada por el usuario. Este hash se realizará mediante la función SHA-1.	La función de este campo será la de poder reconocer en cada momento el usuario que ha iniciado sesión en la aplicación. De esta forma, se podrá usar su valor para, entre otras cosas, recuperar información

		de la base de datos.
SessionId	Posee un tamaño de 10 caracteres. Su contenido será una cadena numérica generada aleatoriamente. Para generar esta cadena, se usarán funciones con técnicas "seguras" que evitarán en mayor medida las colisiones.	Este campo recibirá un valor en el momento de enviar el primer mensaje del protocolo realizado para obtener los datos de perfil. Este valor tendrá validez desde el momento del login hasta que el usuario abandone la aplicación, y será usado para que el servidor pueda conocer con qué sesión intercambia información en cada momento.
Code	Su contenido será una cadena numérica generada aleatoriamente. Para generar esta cadena, se usarán funciones con técnicas "seguras" que evitarán en mayor medida las colisiones.	Este código lo establecen entre las entidades *IdP* o *Host* con su correspondiente *AM*. Es la prueba de que ambas entidades han establecido una conexión previa.

Tabla 3: Campos fijos de los mensajes del protocolo

En la siguiente tabla se mostrarán los campos variables. Con el fin de conocer qué mensajes incluirán cada uno de los campos, se expondrá una segunda tabla en la que aparecerán los números usados como identificador de mensajes para cada una de las partes del protocolo.

Campo	Descripción	Funcionalidad
Location	Este campo almacenará una URL, concretamente la que identificará la posición de los distintos servidores.	En muchos casos será necesario incluir la dirección del servidor con el cual alguna entidad deberá contactar en posteriores mensajes, ya que en principio la desconocerá. Una vez se recibe, se puede proceder con el envío de los siguientes mensajes.
Date	Fecha actual del sistema con el formato: dd-MM-yyyy HH:mm:ss	La fecha será una de las dos partes del mensaje que se firmará, por tanto será necesario proporcionársela a la entidad que vaya a verificar esta. Por otra parte, puede ser de utilidad en futuras mejoras o para realizar logs, conociendo gracias a esta el momento exacto en el que se envía un mensaje.
Signed Message	Para generar su contenido, habrá que concatenar el hash del email del usuario con la fecha del sistema. Toda esta cadena resultante se firmará usando RSA.	La funcionalidad de este campo es simple, ya que se usará exclusivamente para realizar la verificación de la firma de la entidad que proporcione este mensaje.
Token	Se tratará de una cadena aleatoria	Representará el "ticket" proporcionado

	de 5 dígitos.	por parte de un *Authentication Manager* a un usuario, y que le servirá a este último para identificarse frente al *Identity Provider* o al *Host*.
Expires in	El valor de este campo se establecerá por defecto a "7200", el cual representa en segundos el tiempo de validez del token al que acompaña.	La función de este campo es la de indicar el tiempo de validez del token, tras el cual se deberá finalizar la sesión. En esta implementación su valor será "7200 s", dado que se estima que los usuarios no pasan más de dos horas seguidas conectados a las redes sociales. De igual manera, su valor se podrá modificar posteriormente.
Token type	El valor de este campo se establecerá por defecto a "u+f", el cual representa el tipo de token que se proporciona.	La función de este campo es la de permitir a las redes sociales conocer el tipo de token que se les ha proporcionado, para poder establecer la forma de tratarlos. El valor usado por defecto en esta implementación se podrá modificar en un futuro.
Foaf	El campo contendrá un fichero con formato FOAF, el cual ha sido elegido para representar la información de perfil de los usuarios.	Este campo incluye la información del perfil del usuario que la red social solicitó en el primer mensaje.
Key	El campo contiene una clave pública para realizar cifrado.	Esta clave será usada en el *Host*, dado que con ella se cifrarán las claves simétricas con las que se cifraron los recursos.
Resource	Array de bytes que representa los recursos solicitados del usuario. En esta implementación estos recursos serán únicamente fotos, pero se podrá mejorar con el fin de aceptar otro tipo de recursos.	Este campo incluye los recursos del usuario solicitados por la red social. Aparecerá repetido en un mismo mensaje tantas veces como recursos se hayan solicitado.
Age	Campo formado por dos dígitos que representa la edad del usuario que intenta acceder a información de un contacto	Será utilizado para comprobar que cumple las políticas de seguridad establecidas por sus contact5os para el acceso a la información.
School	Texto que indica el centro de estudios del que forma parte el usuario que intenta acceder a información de un contacto.	Será utilizado para comprobar que cumple las políticas de seguridad establecidas por sus contact5os para el acceso a la información.

Tabla 4: Campos variables de los mensajes del protocolo

Campo	Protocolo para la obtención de información del propio usuario	Protocolo para la obtención de datos de un contacto
Location	2, 4,	4, 5, 6, 7, 8, 9, 18
Date	6, 7	1, 11, 12, 13, 15, 16, 17, 18
Signed Message	6, 7	1, 11, 12, 13, 15, 16, 17, 18
Token	9, 10, 11	20, 21, 22, 23, 24
Expires in	9	20, 21
Token type	9	20, 21
Foaf	12 (campo solo posible en la solicitud de perfil)	25, 26 (campo solo posible en la solicitud de perfil)
Key	10 (campo solo posible en la solicitud de recursos)	22, 23 (campo solo posible en la solicitud de recursos)
IV	10 (campo solo posible en la solicitud de recursos)	22, 23 (campo solo posible en la solicitud de recursos)
Resource	12 (campo solo posible en la solicitud de recursos)	25, 26 (campo solo posible en la solicitud de recursos)
Age		17, 18
School		17, 18

Tabla 5: Relación de vampos variables con los mensajes en los que aparecen

2.1.2 Minimización de datos expuestos. Aspectos de seguridad complementarios

En esta sección se tratarán los aspectos de seguridad que habrá que tener en cuenta a la hora de desarrollar el proyecto. Cada aspecto se describe en las siguientes secciones.

En esta sección se trata, en primer lugar, como se va a abordar el objetivo de minimizar los datos expuestos. Posteriormente, se detallan otros aspectos de seguridad que van a ayudar a proteger la información de los usuarios.

2.1.2.1 Minimización de los datos expuestos

Como se ha comentado en la Sección 1.3.2, el protocolo usado persigue el objetivo de minimizar los datos expuestos por los usuarios. Si bien con este protocolo se consigue evitar que las redes sociales tengan el control sobre los datos de los usuarios, aún queda otro problema por resolver.

El problema surge porque, aunque la información no la guarde la red social, habrá un momento en que tendrá que llegar a ésta, para que se pueda mostrar al usuario. En este momento la red social podría de alguna manera almacenar la información, haciendo que el objetivo no se cumpliese.

Para evitar esta situación, se deben aplicar técnicas de cifrado sobre los recursos de los usuarios, de manera que si la red social decidiese almacenar los recursos que le llegan para mostrar a través de su interfaz, no pudiera interpretarlos. Para que esta técnica sea efectiva y el usuario pueda visualizar sus recursos de manera correcta evitando la intromisión de la red social, se debe realizar el cifrado en el propio equipo del usuario, a través de alguna herramienta como un plug-in incorporado al navegador.

El proceso de cifrado de los recursos se llevará a cabo en dos fases, tal y como se observa en la Ilustración XX:

- En primer lugar se cifrarán los recursos en el momento de subirlos a la base de datos. Este primer cifrado será un cifrado simétrico, por lo que se hará uso de una misma clave para cifrar y descifrar. Para que posteriormente se puedan descifrar los recursos y mostrarlos en la interfaz de la red social, se debe subir a la base de datos la clave con la que se haya realizado el cifrado.

- En segundo lugar, se aplicará cifrado asimétrico en el momento en el que uno de los contactos de este usuario que contiene recursos almacenados en la base de datos desea visionarlos. El contacto mandará al *Host* su clave pública, con la que se cifrará la clave contenida en la base de datos. De esta forma, si un atacante obtiene el mensaje, no podrá realizar el descifrado del recurso puesto que su clave también se encuentra cifrada.

  Una vez el mensaje llegase al contacto, éste descifraría la clave simétrica con su clave privada, y acto seguido descifraría el recurso con la clave simétrica recién descifrada.

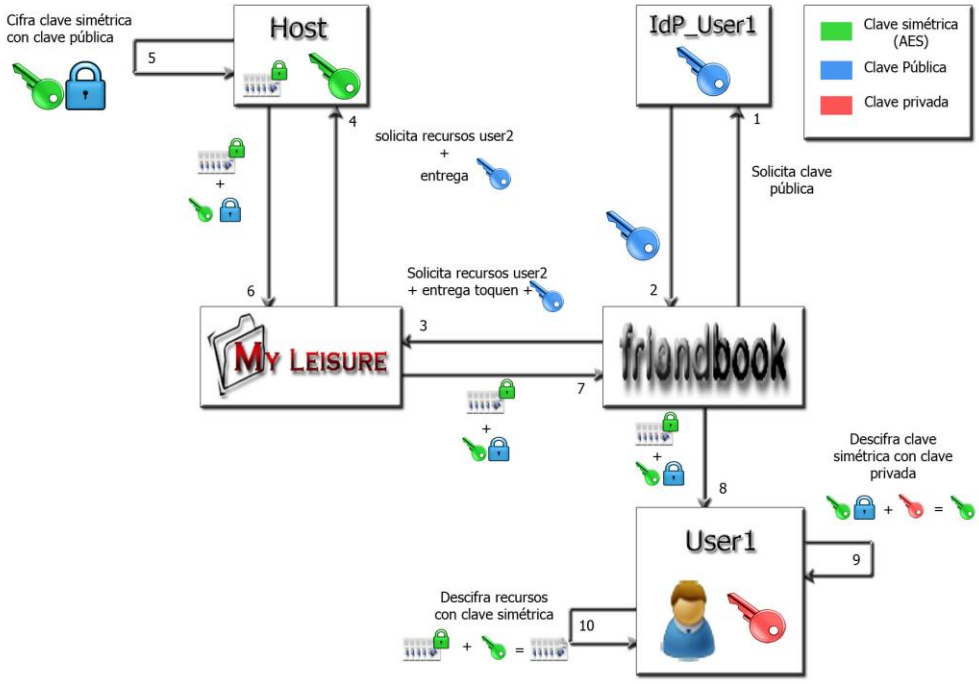

Ilustración 12: Cifrado de recursos

Para finalizar con la explicación del cifrado, comentar que los algoritmo que se usarán por defecto serán "AES", con una clave de 128 bits, en el cifrado simétrico, y RSA, con clave de 2048 bits, en el cifrado asimétrico. Ambos algoritmos se podrán cambiar posteriormente.

2.1.2.2 Uso de "password" para la identificación del usuario

La primera medida de seguridad complementaria al cifrado que aparecerá en la aplicación es la "password" o contraseña, gracias a la cual se podrá verificar que la persona que introduce una dirección de correo electrónico con el objetivo de acceder a su información y la de sus contactos es realmente la propietaria de dicho correo. Pese a que actualmente en las redes sociales los usuarios se autentican con una contraseña (lo cual es la implementación realizada en este proyecto), lo que se pretende es que, además, los usuarios se autentiquen cada vez que comienzan la sesión con su IdP, Host y AM. De esta forma se evitaría que la propia red social suplante su identidad. Muchas son las posibles formas de autenticación (por ejemplo haciendo uso de tarjetas identificadoras, reconocimiento de huellas dactilares…) pero, por simplicidad, se indica la autenticación por medio de algo conocido, de una contraseña.

Para que la contraseña sea fiable, deberá cumplir ciertos requisitos:

- Su longitud deberá ser superior a ocho caracteres, dado que una longitud menor reduciría el esfuerzo necesario para obtener el acceso a la información mediante un ataque de diccionario.
- Con el mismo objetivo, entre estos nueve o más caracteres deberán aparecer: letras minúsculas, letras mayúsculas y algún carácter extraño (por ejemplo "!").

Dada la importancia de las contraseñas en el proceso de autenticación, es inaceptable almacenarlas en claro en la base de datos, dado que se podrían recuperar fácilmente una vez se accediese a ésta.

Para evitar este problema, tan solo se guardará un hash de la contraseña obtenido mediante el algoritmo asimétrico SHA-1, con lo cual no se podrá obtener la contraseña a partir del hash.

2.1.2.3 Firma digital

Este mecanismo de seguridad será muy útil ya que aporta al protocolo las siguientes funcionalidades:

- **Autenticación**: cualquier entidad que posea el certificado público de otra será capaz de autenticarla. Esto se debe a que la firma se realizará mediante el certificado privado, el cual tan solo poseerá la entidad firmante.

- **Integridad**: si un mensaje ha sido alterado, la verificación de la firma no será correcta

- **No repudio**: no se podrá negar el envío de un mensaje por parte de una entidad, ya que el hecho de que aparezca la firma corrobora la participación de ésta.

En este proyecto, la firma digital se usará con el objetivo de identificar la fuente de la que proviene un mensaje, evitando posibles suplantaciones que pudiera realizar un atacante externo para obtener información, o incluso una de las redes sociales que hagan uso del protocolo, para lo cual suplantaría la identidad del *Identity Provider* frente al *Authentication Manager* para obtener un "token".

Las entidades que realizarán firma en alguno de los mensajes que emitan serán: los *Identity Provider*, con el fin de que los *Authentication Manager* puedan asegurar que el sistema que solicita autenticación no ha sido suplantado, y la red social en la que el usuario ha realizado el login. En este segundo caso, la firma se realiza con el objetivo de que la segunda red social pueda verificar que quien le solicita los datos es una red social real a la que puede cedérselos.

Para realizar la firma correctamente, en primer lugar se determinará (en el propio código), el algoritmo que se desea usar, habiéndose establecido por defecto el RSA. A continuación se realizará la firma con la clave privada de la entidad, de manera que nadie más que ella pueda realizar esta firma.

En el momento de la verificación, cabe destacar que ésta se realizará con la clave pública de la entidad firmante, por lo que toda entidad que vaya a realizar verificaciones de firmas deberá poseer previamente el certificado público de la otra entidad (los *AM* siempre tendrán almacenados los certificados públicos de los *IdP*, mientras que las redes sociales pueden determinar con qué otras redes quieren comunicarse, almacenando sólo el certificado de estas), y comprobar que este certificado no ha expirado ni se haya revocado.

El proceso de verificación de firma será sencillo si se tienen en cuenta los detalles descritos en el anterior párrafo. Tan solo se deberá realizar el descifrado del mensaje y comprobar que el resultado obtenido coincide con la cadena resultante de concatenar el hash de la dirección de correo del usuario con la fecha del sistema recibida en el propio mensaje.

2.2 Arquitectura inicial

En esta sección se realizará una descripción de la arquitectura de la que se hará uso en el proyecto. Esta arquitectura se tomará como base para realizar el diseño detallado de la arquitectura final.

Como ya se ha comentado en anteriores secciones, para obtener la funcionalidad que aporta el protocolo será necesario hacer usos de distintos servidores, cada uno con unas tareas específicas: *IdP* proporciona los datos del perfil de un usuario, *Host* proporcionará los recursos de éste, y *AM* que será el encargado de verificar las autenticaciones. Estos servidores se unirán a aquellos que contengan cada una de las redes sociales usadas, para completar la arquitectura.

Para que estos servidores puedan cumplir correctamente las especificaciones del protocolo, deberán ser capaces de almacenar la información que se solicita (ficheros FOAF con los perfiles de los usuarios, fotografías…), así como otros elementos necesarios para el correcto cumplimiento de todas las funcionalidades (certificados públicos y privados, políticas de seguridad…). Con este fin, se crearán múltiples bases de datos, cada una de ellas asociadas a un tipo de servidor diferente, con el fin de que éstos tengan un lugar donde almacenar los datos correspondientes. Por lo tanto, existirán principalmente tres tipos de bases de datos distintas, las ligadas a los *IdP* cuyo contenido más importante serán los perfiles de los usuarios, las ligadas a los *Host* que contendrán todos los recursos de los usuarios, y las que lo estén a los *AM*, que entre otros datos almacenarán las políticas de seguridad establecidas por los usuarios.

Cada una de las bases de datos descritas serán accedidas únicamente desde su servidor correspondiente (por ejemplo, el *Host* será la única entidad que podrá acceder a la base de datos que contiene los recursos). Por tanto, cada base de datos puede limitar el acceso a ésta única entidad, evitando en gran medida ataques externos.

Por último destacar que las redes sociales necesitarán igualmente una base de datos, con el objetivo de almacenar los hash de las cuentas de correo de cada usuario junto a su correspondiente contraseña, así como los enlaces a los servidores de los que hacen uso estos usuarios (se deberán indicar en el registro el *IdP* y el *Host* que almacenarán los datos correspondientes, pudiéndolos modificar en un futuro). En futuras modificaciones, se podría establecer que estos enlaces no se guardasen en la base de datos de las redes sociales, sino que los aportase el usuario en el momento del login a través de ficheros de configuración.

Conocidos ya todos los elementos que se encontrarán en el proyecto y sus respectivas funcionalidades, es momento de establecer como estos estarán intercomunicados con el fin de poder enviarse y recibir los mensajes necesarios para cumplir el protocolo.

El "centro" principal de la arquitectura serán las redes sociales, ya que interactuarán con todo el resto de servidores, mientras que estos, en cambio, no interactuarán todos entre sí.

Cuando el objetivo que se persigue sea el de obtener los datos del perfil de un usuario, la red social interactuará con el *IdP* y su correspondiente *AM*, de la misma forma que lo hará con el *Host* y su *AM* cuando se busquen los recursos de un usuario. En el primero de los procesos *IdP* y *AM_IdP* también interactuarán entre sí para validar el "token" entregado al usuario, y *Host* y *AM_Host* actuarán igual. Por el contrario, no existirá en ningún momento una comunicación entre el proveedor de identificación y el de recursos, ni entre sus correspondientes autenticadores, de ahí que se destaque que no habrá una comunicación de todos con todos.

Es importante destacar que en un principio, la red social no conoce la ubicación de los *AM*, por lo que no puede contactar con ellos hasta que el *IdP* o el *Host* correspondiente le facilite esta ubicación (por este motivo la conexión *SN* ➔ *AM* se representa mediante una línea discontinua).

En la siguiente imagen se podrá observar como quedarían todos los servidores con sus correspondientes comunicaciones entre sí y con las bases de datos, formando una especie de nodo.

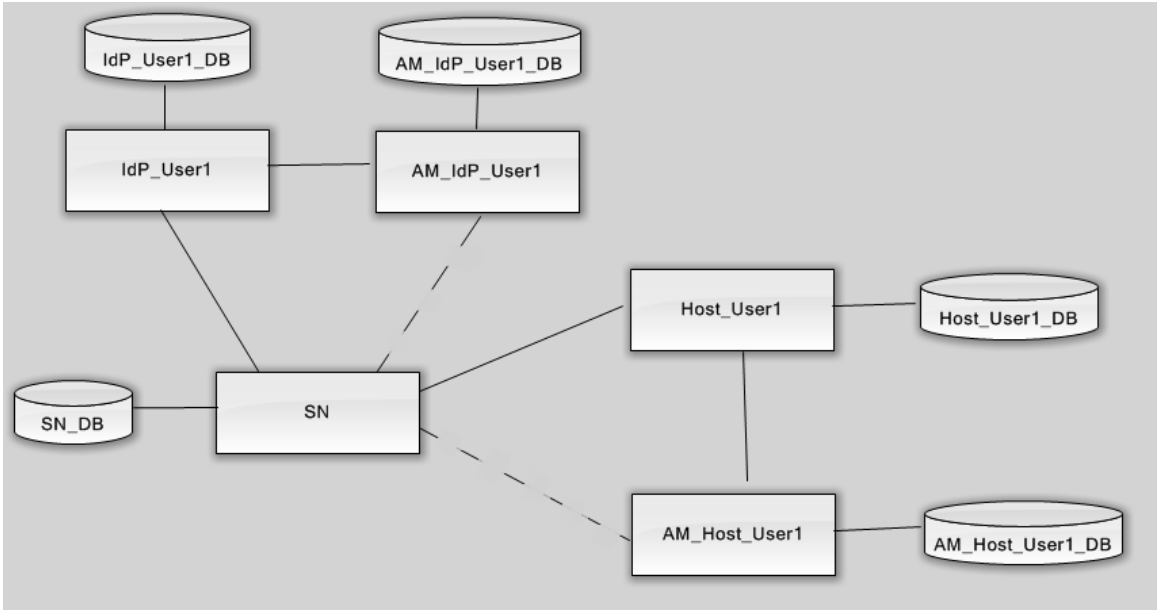

Ilustración 13: Arquitectura inicial de una red social

El hecho de que se llame "nodo" a esta pequeña arquitectura formada por los servidores y sus bases de datos se debe principalmente al aspecto que tomará la arquitectura final en el momento de escalar el sistema. Para realizar esta escala, tan solo se deberá obtener otro "nodo" con los mismos componentes, que se enlazará con el primero a través de una conexión entre las dos redes sociales.

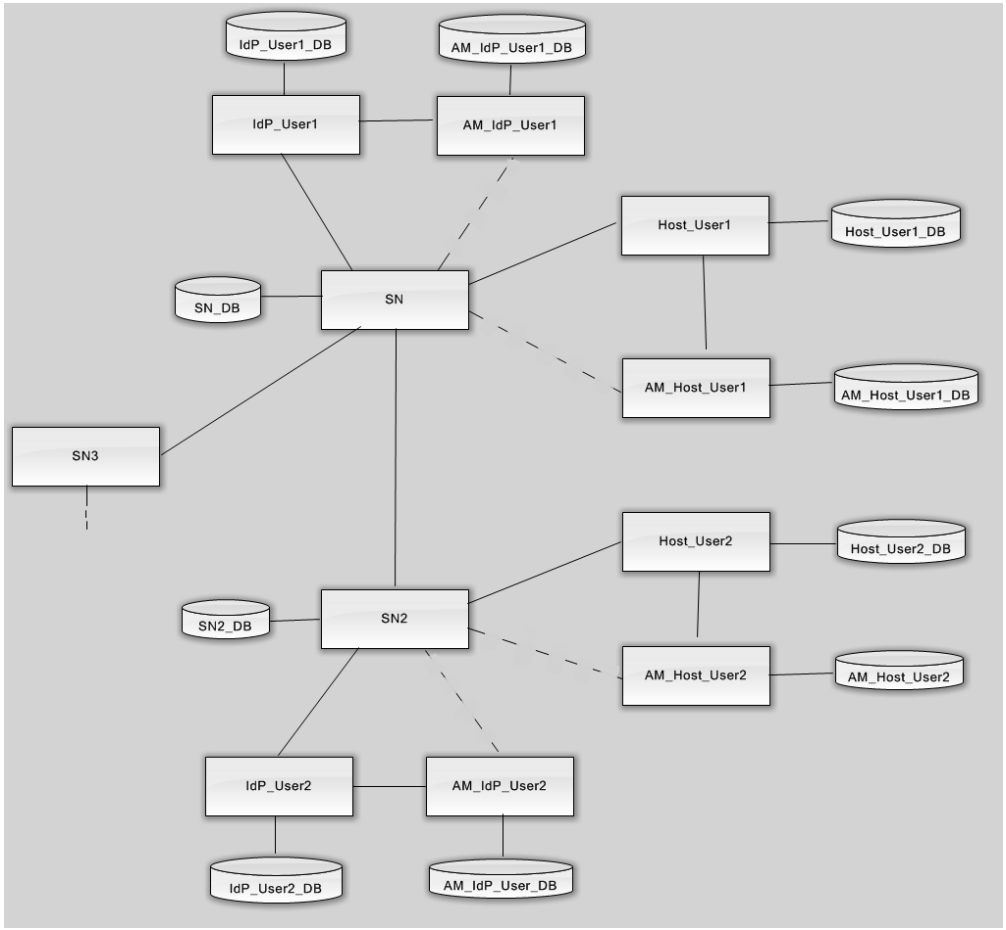

Ilustración 14: Arquitectura inicial de múltiples redes sociales

Este hecho propicia que se pueda definir este sistema como un sistema fácilmente escalable, ya que cada vez que se quiera incluir a él una nueva red social, tan solo se deberán establecer conexiones con las redes sociales oportunas.

2.3 Descripción de las tecnologías usadas

En esta sección se describen cada una de las tecnologías usadas para el desarrollo de la aplicación. Además, se analizarán brevemente las alternativas posibles a las tecnologías usadas y se expondrá el o los motivos por los que se ha decidido finalmente el uso de una de ellas.

2.3.1 JAVA EE

Plataforma de programación para el desarrollo de software en lenguaje JAVA y su posterior ejecución sobre un servidor de aplicaciones. Posee varias especificaciones API, de las cuales caben destacar "servlets", que se usarán como parte controladora de la aplicación, y JDBC, para realizar conexiones y consultas con las bases de datos, y Java Server Pages (JSP), la cual será esencial para representar cada una de las páginas que compondrán la aplicación web.

Otras ventajas que aporta esta tecnología es su portabilidad, pudiéndose ejecutar en máquinas distintas gracias a la Java Virtual Machine. También es destacable la posibilidad de integración con otras tecnologías, y su facilidad para escalar, así como el API que provee, el cual incluye, entre otras, funciones para la seguridad de la información, aspecto muy importante para este proyecto.

La situación actual de las tecnologías ofrece como alternativa a JAVA EE la plataforma .NET. La decisión de optar por JAVA EE como plataforma usada para el desarrollo de la aplicación se debe, por un lado, al abanico de entornos de programación que ofrece esta tecnología (NetBeans, Eclipse, Visual Studio…) frente a la única posibilidad de hacer uso de Visual Studio para .Net. Por otro lado, el hecho de ejecutar sobre la Java Virtual Machine, lo que ofrece la posibilidad de ejecutar casi sobre cualquier plataforma. Por último, otro motivo de decisión ha sido la amplia experiencia del desarrollador en esta tecnología respecto a otras.

2.3.2 NetBeans 7

NetBeans es un entorno de desarrollo integrado, disponible para sistemas operativos Windows, Mac, Linux y Solaris. Se trata de un software de código abierto que permite el desarrollo de aplicaciones web, de escritorio o para dispositivos móviles.

Este entorno permite hacer uso de las plataformas JAVA, PHP, JavaScript, AJAX, Groovy and Grails y C/C++.

Como alternativas a NetBeans, se encuentra principalmente el entorno de programación integrado Eclipse, también de código abierto y multiplataforma.

Dado que ambos ofrecen características similares, el motivo que ha llevado a la elección final de NetBeans ha sido la incorporación y configuración por defecto de los servidores de aplicaciones Apache Tomcat y GlassFish, lo que ofrece una mayor sencillez a la hora de montar el entorno de trabajo.

2.3.3 GlassFish v3

Servidor de aplicaciones elegido para ejecutar tanto las redes sociales como los distintos servidores que aparecen en la aplicación.

Al igual que se ha elegido GlassFish, se podría haber optado por el contenedor de aplicaciones por todos conocido Apache Tomcat, ya que ambos son de código abierto y dan un gran rendimiento, pero se ha optado por GlassFish por su menor complejidad y su perfecta integración con NetBeans.

2.3.4 MySQL

MySQl es un sistema de gestión de bases de datos relacional, perteneciente a la compañía Oracle Corporation, con más de seis millones de instalaciones (14).

MySQL se puede usar sin ningún coste, lo cual lo posiciona por delante de otros gestores que aumentarían el coste del proyecto. Otras características que han llevado a su uso son su escalabilidad y fiabilidad (permite administrar hasta terabytes de información), su alto rendimiento, y su fuerte protección de datos, aspecto muy importante teniendo en cuenta la sensibilidad de la información que manejará la aplicación.

2.3.5 GSON

Se trata de una librería para JAVA que aporta la funcionalidad de convertir objetos al formato JSON y viceversa.

GSON es un “open-source”, por lo que no supone ningún coste al proyecto.

La característica que diferencia a esta librería ofrecida por Google de otras como FLEXJSON o JSON-lib, que también trabajan con el formato JSON, es que no requiere ningún tipo de anotación especial en el código JAVA.

2.4 Arquitectura definitiva

En esta sección se va a describir la arquitectura definitiva a utilizar en el proyecto, la cual parte de la inicial, detallando algunos aspectos relevantes y mencionando posibles cambios que puedan surgir de restricciones causadas por las tecnologías usadas.

En primer lugar se ha de destacar que las tecnologías escogidas se ajustan perfectamente a las necesidades de la arquitectura inicial, por lo que no es necesaria ninguna modificación a causa de éstas.

Profundizando en la arquitectura que se usará en el proyecto, se ha de destacar en primer lugar que seguirá el esquema MVC (Modelo – Vista - Controlador), dado que existirá una clara diferenciación entre la parte de la interfaz mostrada al usuario (proporcionada por cada una de las redes sociales), el modelo de datos (que se encontrará repartido en diferentes bases de datos asociadas a los servidores) y el controlador, representado por todos los servlets que permiten la intercomunicación entre los servidores y la red social.

El uso de este patrón facilita las posibles modificaciones y mejoras que se deseasen realizar en un futuro sobre la aplicación. Llegado este momento, en primer lugar habría que determinar que elemento es el que se desea modificar: la interfaz de las redes sociales, el tipo de datos que se almacenan o la forma de hacerlo, o la lógica de negocio (por ejemplo, si se desease hacer uso de un único *AM* en lugar de dos).

Por otro lado, se ha de tener en cuenta que se van a usar módulos a parte de las entidades, que proporcionarán, en uno de los casos, acceso a las bases de datos y la posibilidad de realizar sobre estas consultas y modificaciones. En el otro caso, se tratará de un módulo que aportará la criptografía, es decir, las funciones de las que se harán uso para realizar firmas, cifrado, funciones hash… Estos módulos, llamados desde las entidades que los requiriesen, simplifican el código de las entidades, haciéndolo más comprensible para futuras modificaciones. Además, permiten modificaciones más rápidas, como cambios en los algoritmos de seguridad usados, o cambios en las tablas de las bases de datos.

En el siguiente diagrama se pueden observar los componentes que conforman el sistema y como se interconectan.

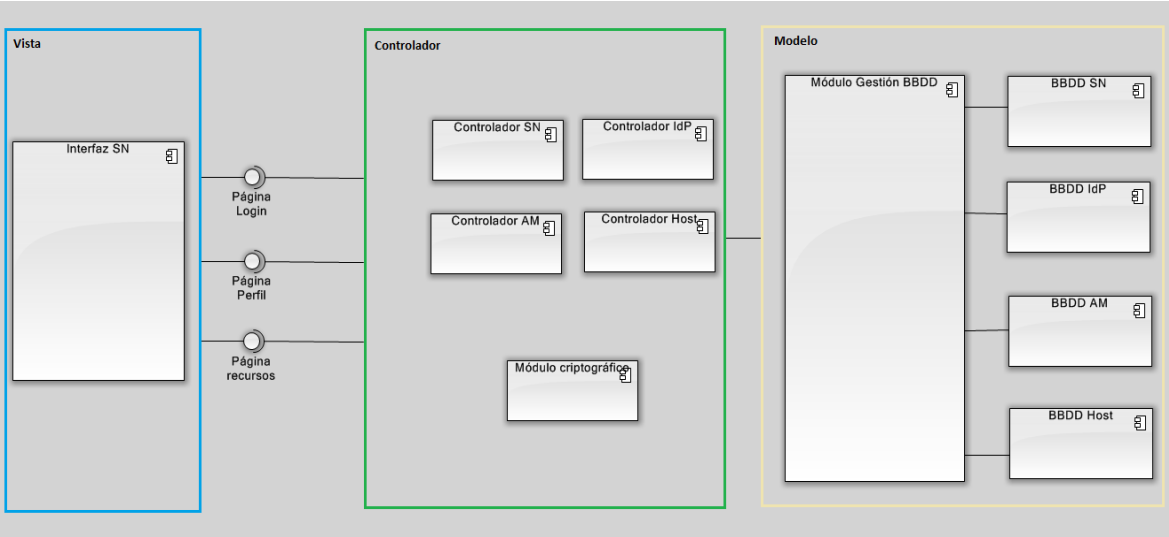

Ilustración 15: Arquitectura definitiva

Es necesario detallar algunos aspectos para mejorar la comprensión del diagrama. En primer lugar, comprobar como todos los controladores conectan con el módulo de gestión de las bases de datos (líneas rojas), así como con el módulo criptográfico (líneas verdes). Esto denota la gran importancia de ambos módulos, ya que son constantemente usados por todos los elementos, por tanto su creación evita una gran cantidad de redundancia en el código, obteniendo como resultado un código más simple.

Por otro lado, de la misma forma que se comentó en la arquitectura inicial, destacar que en primera instancia, el controlador de la red social no se encuentra conectado a los controladores de ambos *AMs*. Esta conexión se efectúa una vez que el *IdP* o el *Host* han facilitado la localización de los *AM* a la red social que los requiere.

2.5 Casos de uso

Realizando el análisis de la aplicación y de los requisitos de software, se han extraído los siguientes casos de uso:

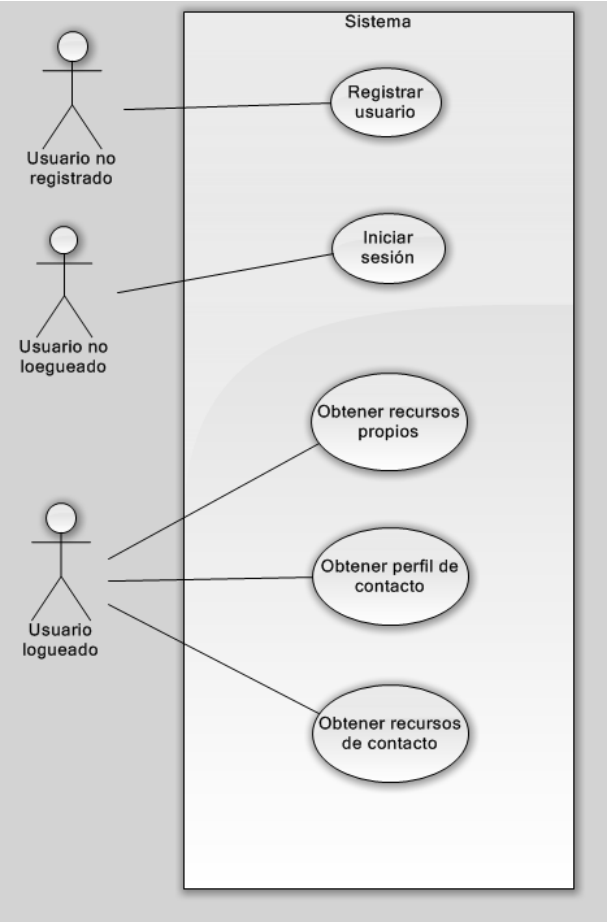

Ilustración 16: Diagrama de casos de uso

Para especificar cada uno de estos casos de uso, se va a utilizar una tabla con el siguiente formato:

Caso de uso:	
Actores:	
Pre-condiciones:	
Escenario:	
Escenario alternativo:	
Post-condiciones:	

Tabla 6: Formato de casos de uso

Con el objetivo de aclarar el contenido de cada campo, a continuación se describirán uno por uno:

- Caso de uso: título descriptivo del caso de uso que se analizará en cada tabla
- Actores: entidades o roles que tomarán parte en el caso de uso
- Pre-condiciones: estado del sistema necesario para la ejecución del caso de uso.
- Escenario: descripción de la operativa que habrá que llevar a cabo para realizar el caso de uso.
- Escenario alternativo: cumple la misma función que el campo "Escenario", pero solo tendrá valor si pueden darse situaciones distintas al escenario principal durante la realización del caso de uso.
- Post-condiciones: estado del sistema tras la ejecución del caso de uso.

Conocidos tanto los distintos casos de uso que aparecerán en el sistema, como el formato con el que se presentará la especificación de estos, se procede a su descripción.

Caso de uso:	**Registrar usuario**
Actores:	Usuario no registrado
Pre-condiciones:	• Red social arrancada
Escenario:	• Pulsar sobre el botón "Registrar" • Rellenar los campos solicitados, incluyendo las direcciones de *IdP* y *Host* • Pulsar el botón "Finalizar registro"
Escenario alternativo:	• Pulsar sobre el botón "Registrar" • Rellenar los campos solicitados, incluyendo las direcciones de *IdP* y *Host* • Pulsar sobre el botón "Finalizar registro" • Comprobar los campos que han sido erróneos y modificarlos para que cumplan las condiciones • Pulsar sobre el botón "Finalizar registro"
Post-condiciones:	• Información de registro almacenada en el *Identity Provider (IdP)*

Tabla 7: Caso de uso – Registrar usuario

Caso de uso:	**Iniciar sesión**
Actores:	Usuario registrado
Pre-condiciones:	• Red social arrancada y situada en la página del cuadro de login.
Escenario:	• Introducir dirección de correo electrónico • Introducir contraseña • Pulsar el botón “Iniciar sesión”
Escenario alternativo:	N/A
Post-condiciones:	• Acceso permitido: página de perfil con los datos del usuario • Acceso denegado: página de login con mensaje de error en el inicio de la sesión.

Tabla 8: Caso de uso – Iniciar sesión

Caso de uso:	**Obtener recursos propios**
Actores:	Usuario con sesión iniciada
Pre-condiciones:	• Red social en la página de perfil, con la sesión iniciada
Escenario:	• Pulsar el botón “Fotos” de la barra izquierda
Escenario alternativo:	N/A
Post-condiciones:	• Acceso permitido: página con las fotos del usuario • Acceso denegado: mensaje de error indicando el problema sucedido

Tabla 9: Caso de uso – Obtener recursos propios

Caso de uso:	**Obtener perfil de contacto**
Actores:	Usuario con sesión iniciada
Pre-condiciones:	• Red social en la página de perfil, con la sesión iniciada y uno o más contactos añadidos.
Escenario:	• Seleccionar el contacto deseado e la lista de contactos • Seleccionar la red social de la que se desea extraer los datos
Escenario alternativo:	• Seleccionar el contacto deseado e la lista de contactos • Seleccionar la red social de la que se desea extraer los datos • Obtener un mensaje de error • Solicitar los datos a otra de sus redes sociales
Post-condiciones:	• Acceso permitido: página de perfil con los datos del contacto • Acceso denegado: mensaje de error indicando el problema sucedido

Tabla 10: Caso de uso – Obtener perfil de contacto

Caso de uso:	**Obtener recursos de contacto**
Actores:	Usuario con sesión iniciada
Pre-condiciones:	• Red social en la página de perfil, habiendo realizado la carga de datos de un contacto
Escenario:	• Seleccionar el contacto deseado e la lista de contactos • Seleccionar la red social de la que se desea extraer los datos • Pulsar el botón “Fotos” de la barra izquierda
Escenario alternativo:	• Seleccionar el contacto deseado e la lista de contactos • Seleccionar la red social de la que se desea extraer los datos • Obtener un mensaje de error • Solicitar los datos a otra de sus redes sociales • Pulsar el botón “Fotos” de la barra izquierda
Post-condiciones:	• Acceso permitido: página con las fotografías del contacto • Acceso denegado: mensaje de error indicando el problema sucedido

Tabla 11: Obtener recursos de contacto

2.6 Requisitos del software

Esta sección será dedicada a la exposición de los requisitos de software que se han determinado que deberá cumplir la aplicación a desarrollar, con el fin de que ésta cumpla todas las funcionalidades definidas por el cliente.

La especificación de requisitos se realiza siguiendo la metodología ESA (15), de la que se extraen las categorías en las que se dividen los requisitos y los campos que se deben establecer para cada requisito.

Con el fin de comprender las características de los requisitos, previa a su exposición se explicará el formato con el que se presentarán cada uno de ellos.

2.6.1 Formato de requisitos

Todos los requisitos se presentarán haciendo uso de una tabla como la siguiente:

Tipo de requisitos				
Id.	**Versión**	**Descripción**	**Prioridad**	**Necesidad**

Tabla 12: Formato de requisitos

A continuación se detallará la función de cada uno de los campos:

- **Identificador del requisito:** estará formado por dos letras que indicarán el tipo de requisito que se trata (RF (Requisito funcional) o RNF (Requisito no funcional)) junto a un número que lo identifique y distinga de otros requisitos de su mismo tipo. Por lo tanto, un ejemplo del formato final sería: "RF-01".

  El objetivo de este identificador es el poder distinguir cada uno de los requisitos, lo cual será muy útil a la hora de realizar un estudio que certifique que el "Plan de pruebas" verifica la funcionalidad de cada uno de ellos.

- **Versión:** indicará la versión en la que se encuentra el requisito, en el caso de que sufriera modificaciones.

- **Descripción:** indica la funcionalidad que cubre el requisito.

- **Prioridad:** su valor podrá ser "Alta", "Media" o "Baja". Cuanto mayor sea la prioridad, mayor preferencia tendrá el requisito frente a otro.

- **Necesidad:** su valor podrá ser "Alta", "Media" o "Baja". Un grado más alto de necesidad indica que es más importante su cumplimiento para el sistema, ya sea por su relación con otros requisitos, por tratarse de una funcionalidad muy importante, o cualquier otro motivo.

2.6.2 Requisitos funcionales

Los requisitos funcionales son aquellos que representan las funcionalidades que deberá cumplir la aplicación a desarrollar. A continuación se mostrarán todos los requisitos funcionales.

Requisitos funcionales					
Id.	**Versión**	**Título**	**Descripción**	**Prioridad**	**Necesidad**
RF-01	1.0	Registro de usuarios	La aplicación permitirá el registro de usuarios nuevos	MEDIA	MEDIA
RF-02	1.0	Solicitud de servidores externos en registro	La aplicación solicitará las direcciones de los servidores *IdP* y *Host* en el momento del registro de un nuevo usuario	ALTA	ALTA
RF-03	1.0	Modificación de servidores externos	La aplicación ofrecerá la posibilidad de modificar las direcciones de los servidores *IdP* y *Host* de manera posterior al registro	MEDIA	MEDIA
RF-04	1.0	Inicio de sesión	La aplicación permitirá a los usuarios iniciar sesión	ALTA	ALTA
RF-05	1.0	Requisitos para inicio de sesión	La aplicación solicitará al usuario su dirección de correo electrónico y su contraseña para iniciar sesión	ALTA	ALTA
RF-06	1.0	Alerta para servidores externos	La red social alertará a los servidores *IdP* y *Host* que un usuario que posee información contenida en ellos ha realizado el login	ALTA	ALTA
RF-07	1.0	Conocimiento de *AMs por* IdPs	Los servidores *IdP* deberán conocer la dirección de los servidores *AM_IdP* que usarán para verificar las autenticaciones	ALTA	ALTA
RF-08	1.0	Conocimiento de *AMs* por *Host*	Los servidores *Host* deberán conocer la dirección de los servidores *AM_Host* que usarán para verificar las autenticaciones	ALTA	ALTA
RF-09	1.0	Creación de nueva sesión	Durante el proceso de inicio de sesión, se creará una sesión específica para el usuario	ALTA	ALTA
RF-10	1.0	Obtención de perfil de usuario	La aplicación ofrecerá la información personal y la lista de contactos al usuario una vez inicie la sesión	ALTA	ALTA
RF-11	1.0	Obtención de recursos de usuario	La aplicación ofrecerá al usuario la posibilidad de recuperar sus recursos	ALTA	ALTA

RF-12	1.0	Obtención de redes sociales de un contacto	La aplicación ofrecerá al usuario la posibilidad de conocer las redes sociales de cada uno de sus contactos.	ALTA	ALTA
RF-13	1.0	Obtención del perfil de un contacto	La aplicación ofrecerá al usuario la posibilidad de recuperar los datos del perfil de un contacto contenidos en una red social diferente.	ALTA	ALTA
RF-14	1.0	Obtención de recursos de un contacto	La aplicación ofrecerá al usuario la posibilidad de recuperar los recursos de un contacto contenidos en una red social diferente.	ALTA	ALTA
RF-15	1.0	Verificación de políticas de control de acceso	La aplicación comprobará que un usuario que intenta acceder a la información de un segundo cumple las políticas de seguridad determinadas por este segundo	ALTA	ALTA
RF-16	1.0	Establecimiento de políticas de control de acceso	La aplicación permitirá establecer las políticas de control de acceso en función a dos restricciones: rango de edad y centro de estudios del usuario.	ALTA	ALTA
RF-17	1.0	Mensajes GET y POST	Las entidades podrán enviar y recibir mensajes HTTP de tipo GET y de tipo POST	ALTA	ALTA
RF-18	1.0	Extracción de mensajes	Las entidades podrán recuperar cada uno de los campos incluidos en los mensajes	ALTA	ALTA
RF-19	1.0	Verificación mediante “token”	El *IdP* y el *AM_IdP* verificarán la autenticación de un usuario mediante la comprobación del “token”	ALTA	ALTA
RF-20	1.0	Finalización automática de sesión	La aplicación deberá finalizar la sesión automáticamente cuando el período de validez del token expire, siempre que ésta no se haya finalizado antes mediante acción del usuario	ALTA	ALTA
RF-21	1.0	Registro en logs	La aplicación registrará mensajes y errores.	MEDIA	MEDIA

Tabla 13: Requisitos funcionales

2.6.3 Requisitos no funcionales

Descripción de todos los requisitos que deberá cumplir la aplicación, exceptuando aquellos que tratan funcionalidades del sistema, que ya se especificaron en la anterior sección. Entre estos requisitos se encontrarán los de rendimiento (establecen tiempos de ejecución), de interfaz (definen el aspecto de la aplicación) y de seguridad (definen las medidas y mecanismos de seguridad).

Requisitos no funcionales					
Id.	**Versión**	**Título**	**Descripción**	**Prioridad**	**Necesidad**
Requisitos de rendimiento					
RNF-01	1.0	Tiempo aceptable en inicio de sesión	El proceso de login se deberá realizar en un tiempo no superior a los 5 segundos	ALTA	ALTA
RNF-02	1.0	Tiempo aceptable en obtención de recursos de un usuario	El proceso de recuperación de recursos del usuario se deberá realizar en un tiempo no superior a los 10 segundos	ALTA	ALTA
RNF-03	1.0	Tiempo aceptable en obtención del perfil de un contacto	El proceso de recuperación del perfil de un contacto deberá realizarse en un tiempo no superior a los 10 segundos	ALTA	ALTA
RNF-04	1.0	Tiempo aceptable en obtención de recursos de un contacto	El proceso de recuperación de recursos de un contacto deberá realizarse en un tiempo no superior a los 15 segundos	ALTA	ALTA
Requisitos de interfaz					
RNF-05	1.0	Login en página principal	En la página inicial se deberá mostrar un cuadro de login.	ALTA	ALTA
RNF-06	1.0	Campos del cuadro de login	El cuadro de login posibilitará la inserción de una dirección de correo y de una contraseña	ALTA	ALTA
RNF-07	1.0	Ubicación de lista de contactos	En la página de perfil se incluirá la lista de contactos del usuario	ALTA	ALTA
RNF-08	1.0	Uso de scroll en listas	Todas las listas que aparezcan en la aplicación poseerán un scroll que permita deslizarlas verticalmente	ALTA	ALTA
RNF-09	1.0	Botón de retorno al perfil del usuario	Todas las páginas de la aplicación ofrecerán la posibilidad de regresar a la página de perfil del usuario haciendo uso de un botón al efecto	ALTA	ALTA
RNF-10	1.0	Botón de cierre de sesión	Todas las páginas de la aplicación ofrecerán la posibilidad de cerrar la sesión haciendo uso del botón “Cerrar sesión”	ALTA	ALTA

RNF-11	1.0	Indicador de página actual	Cada página de la aplicación deberá indicar mediante un nombre identificativo en qué página se encuentra el usuario (perfil, lista de redes sociales, perfil de un contacto…)	MEDIA	MEDIA
RNF-12	1.0	Colores distintivos	Los colores y estructura de cada red social deberán permitir diferenciarla de la otra.	ALTA	ALTA
RNF-13	1.0	Campo de contraseña oculto	El campo dedicado a la inserción de la contraseña en el cuadro de login no permitirá que se vea en claro el texto que se esté introduciendo, sustituyéndolo por *.	ALTA	ALTA
Requisitos de seguridad					
RNF-14	1.0	Algoritmo utilizado: firmas	Se hará uso del algoritmo RSA para realizar las firmas	ALTA	ALTA
RNF-15	1.0	Algoritmo utilizado: hash	Se hará uso del algoritmo SHA-1 para realizar las funciones resumen	ALTA	ALTA
RNF-16	1.0	Algoritmo utilizado: cifrado	Se hará uso del algoritmo AES para el cifrado simétrico, y de RSA para el asimétrico	ALTA	ALTA
RNF-17	1.0	Limitación de acceso a BBDD	El acceso a las BBDD estará limitado, pudiendo acceder únicamente el servidor ligado a ella.	ALTA	ALTA

Tabla 14: Requisitos no funcionales

2.7 Plan de pruebas

En esta sección se va a proceder a definir el plan de pruebas que habrá que llevar a cabo una vez desarrollado el software para comprobar que todas las funcionalidades descritas en los requisitos se cumplen.

Para ello, se especificará cómo realizar la prueba y qué resultado se ha de obtener para darla como aceptada.

El formato con el que se presentarán todas las pruebas de aceptación del sistema será el siguiente:

Pruebas de aceptación				
Identificador	**Título**	**Modo de realización**	**Salida esperada**	**Requisitos verificados**

Tabla 15: Formato de la definición de pruebas

Destacar que el campo "Identificador" seguirá el mismo formato establecido para los requisitos, es decir, dos siglas que indiquen que se trata de una prueba de aceptación seguido de un número que la identifique. Un ejemplo de identificador de prueba de aceptación sería "PA - 01". Por otro lado, el texto que encontramos como "Prueba de aceptación", será sustituido por el título de la prueba en sí.

Con todas las pruebas descritas, se presentará dos tablas (requisitos funcionales y no funcionales) en las que se presenten los requisitos del sistema a las pruebas de aceptación. En esta tabla se podrá comprobar qué requisitos cubre cada prueba, y verificar que todos los requisitos han sido probados.

Una vez el sistema se encuentra completamente finalizado y desarrollado, se llevan a cabo las pruebas de la Tabla XX. Para verificar que se han probado todas ellas y dejar constancia del resultado obtenido, se completará una tabla como la siguiente:

Verificación de pruebas		
Identificador	**Versión**	**Resultado**

Tabla 16: Formato de la verificación de pruebas

Con los formatos ya descritos, se procede a la presentación de las pruebas de aceptación del sistema:

Pruebas de aceptación				
Identificador	**Título**	**Modo de realización**	**Salida esperada**	**Requisitos verificados**
PA-01	Registro de usuarios	**1.**Acceder a la web de una de las redes sociales **2.**Realizar el registro de un nuevo usuario	- Mensaje de confirmación de que la cuenta ha sido creada	RF-01, RF-02, RF-16
PA-02	Modificación de enlaces de servidores	**1.**Acceder al menú de configuración de cuenta **2.**Modificar las direcciones de los servidores **3.**Confirmar los cambios	- Mensaje de confirmación de cambios	RF-03
PA-03	Inicio de sesión	**1.**Acceder a la web de la red social en la que se ha realizado el registro **2.**Realizar el login con los datos de registro	- Página de perfil con los datos del usuario creado	RF-4, RF-5, RF-6, RF-9, RNF-1, RNF-5, RNF-6, RNF-13, RNF-15
PA-04	Obtención de datos del perfil del propio usuario	**1.**Iniciar sesión en una de las redes sociales	- Página de perfil del usuario con los datos correctos de éste	RF-7, RF-10, RF-17, RF-18, RF-19, RNF-1, RNF-8, RNF-10, RNF-11, RNF-12, RNF-14, RNF-15, RNF-17
PA-05	Obtención de recursos del propio usuario	**1.**Realizar login en una de las redes sociales **2.**Pulsar el botón “Fotos” de la barra izquierda	- Visualización de los recursos del usuario	RF-8, RF-11, RF-15, RF-17, RF-18, RF-19, RNF-2, RNF-8, RNF-9, RNF-10, RNF-11, RNF-12, RNF-14, RNF-15, RNF-16, RNF-17
PA-06	Obtención de datos del perfil de un contacto	**1.**Realizar login en una de las redes sociales **2.**Seleccionar un contacto de la lista de contactos **1.**Seleccionar una de las redes sociales disponibles para éste	- Página de perfil con los datos del contacto	RF-7, RF-12, RF-13, RF-15, RF-17, RF-18, RF-19, RNF-3, RNF-8, RNF-9, RNF-10, RNF-11, RNF-12, RNF-14, RNF-15, RNF-17
PA-07	Obtención de recursos de	**1.**Realizar login en una de las	- Visualización de los recursos	RF-8, RF-14, RF-15, RF-17,

	un contacto	redes sociales **2.**Seleccionar un contacto de la lista de contactos **3.**Seleccionar una de las redes sociales disponibles para éste **4.**Pulsar el botón “Fotos” de la barra izquierda	del contacto	RF-18, RF-19, RNF-4, RNF-8, RNF-8, RNF-10, RNF-11, RNF-12, RNF-14, RNF-15, RNF-16, RNF-17
PA-08	Comprobación de logs	**1.**Acceder a los logs generados por el sistema	- Listado de mensajes que se envían durante el uso de la aplicación - Campos firmados y cifrados de los mensajes	RF-21
PA-09	Fin de sesión automático	**1.**Iniciar sesión en una red social **2.**Esperar las dos horas de validez del token **3.**Intentar realizar alguna acción	- Mensaje de error por fin de sesión	RF-20

Tabla 17: Pruebas de aceptación

	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06	RF-07	RF-08	RF-09	RF-10	RF-11	RF-12	RF-13	RF-14	RF-15	RF-16	RF-17	RF-18	RF-19	RF-20	RF-21
PA-01	✓	✓														✓					
PA-02			✓																		
PA-03				✓	✓	✓			✓												
PA-04							✓			✓							✓	✓	✓		
PA-05								✓			✓				✓		✓	✓	✓		
PA-06							✓					✓	✓		✓		✓	✓	✓		
PA-07								✓						✓	✓		✓	✓	✓		
PA-08																					✓
PA-09																				✓	

Tabla 18: Relación pruebas de aceptación / Requisitos funcionales

	RNF-01	RNF-02	RNF-03	RNF-04	RNF-05	RNF-06	RNF-07	RNF-08	RNF-09	RNF-10	RNF-11	RNF-12	RNF-13	RNF-14	RNF-15	RNF-16	RNF-17
PA-01																	
PA-02																	
PA-03	✓				✓	✓							✓		✓		
PA-04	✓						✓	✓		✓	✓	✓		✓	✓		✓
PA-05		✓						✓	✓	✓	✓	✓		✓	✓	✓	✓
PA-06			✓					✓	✓	✓	✓	✓		✓	✓		✓
PA-07				✓				✓	✓	✓	✓	✓		✓	✓	✓	✓
PA-08																	
PA-09																	

Tabla 19: Relación pruebas de aceptación / Requisitos no funcionales

Capítulo 3

– Diseño –

3.1 Diseño del software

Esta sección explica detalladamente cada uno de los componentes del sistema. La sección se divide en cinco partes en las que se comenta el diseño de las dos redes sociales a implementar (Friendbook y MyLeisure, tal y como se describirá posteriormente) y los servidores *IdP*, *Host*, *AM* y el resto de módulos que componen el sistema.

Para realizar el diseño del software, se seguirá el patrón MVC (*Modelo – Vista - Controlador*), el cual adquiere este nombre por dividir el software en tres partes claramente diferenciadas, tal y como se detalló en la Sección 2.2.

Este patrón se cumple estrictamente en las redes sociales implementadas, pero sólo se cumple parcialmente en el desarrollo de los servidores, dado que estos últimos no contemplan el elemento *Vista* al no ofrecer ningún tipo de interfaz.

El diagrama global y simplificado del sistema que se va a detallar a continuación, puede observarse en la siguiente ilustración.

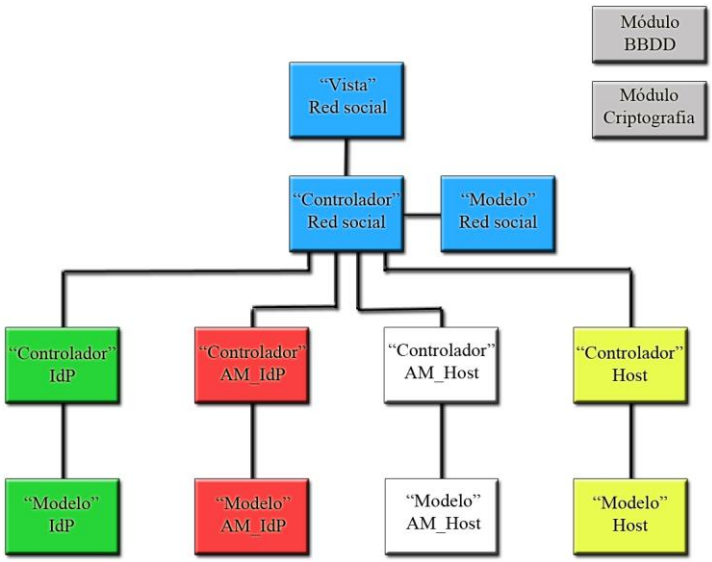

Ilustración 17: Diagrama de componentes simplificado

En este diagrama se pueden diferenciar los elementos en función del servidor al que correspondan. Por un lado se encuentra la red social (azul), único servidor que proporciona el elemento *Vista* al usuario. Junto a ésta, encontramos los cuatro servidores encargados de proporcionar la información: *IdP* (verde), *Host* (amarillo), *AM_IdP* (rojo) y *AM_Host* (blanco). Para finalizar, se encuentran los módulos que facilitarán la labor de todos estos servidores, el módulo criptográfico y el encargado de la gestión de bases de datos (ambos en gris).

A continuación, se van a describir los elementos *Modelo*, *Vista* y *Controlador* de cada una de las entidades participantes. Para facilitar la comprensión de dichas descripciones, se van a presentar los diagramas de clases de cada uno de los elementos.

3.1.1 Redes sociales

Dado que el hecho de probar el protocolo implementado con una red social real (Facebook, Twitter…) no es factible porque requeriría modificarlas (lo cual sólo pueden hacer las propias compañías), se decide implementar dos simulaciones de estas redes sociales con funcionalidades similares (crear un perfil, iniciar sesión, mostrar fotografías…). Estas redes sociales se denominan Friendbook y MyLeisure.

El motivo por el que se diseñan e implementan dos redes sociales distintas es para poder, considerando el objetivo de interoperabilidad, realizar una simulación lo más fiel posible a la realidad. La simulación se podría realizar con una sola interfaz distribuida en dos servidores, pero creando dos interfaces se consigue una mayor claridad a la hora de demostrar el intercambio de datos.

A continuación se detalla el diseño de cada uno de estos tres elementos en las respectivas redes sociales.

3.1.1.1 Vista

En primer lugar se define el componente *Vista*, debido a que al hacer uso de la aplicación, será el primer elemento que actúe de los tres.

Este elemento estará compuesto de cada una de las páginas web que compondrán la red social. Para conocer cuántas páginas deben implementarse y qué funcionalidades deben cumplir, se revisará la definición de casos de uso y de requisitos realizada en el Análisis de la aplicación. De esta forma, se determina que deberán existir suficientes páginas para mostrar: cuadro de login para iniciar la sesión en la aplicación, datos del perfil de los usuarios en los que se incluyen los datos personales, la lista de contactos y las redes sociales en las que estos mantienen un perfil, y los recursos de los usuarios. Además, en futuras mejoras se deben incorporar la interfaz para realizar el registro de un usuario, así como la que permita modificar los datos de éste.

Todas estas páginas se implementarán haciendo uso de JSP, los cuales permiten insertan tanto código HTML como JAVA, este último con el objetivo de realizar operaciones que determinen que campos se pintarán.

A los JSP, se enlazará un fichero CSS, que será el encargado de determinar el formato de cada una de las secciones de las páginas web.

Con el fin de crear una diferencia clara entre ambas redes sociales (Friendbook y MyLeisure), la estructura de páginas cambiará de una a otra. Por este motivo, la estructura de cada una de las redes sociales se definirá individualmente.

3.1.1.1.1 Friendbook

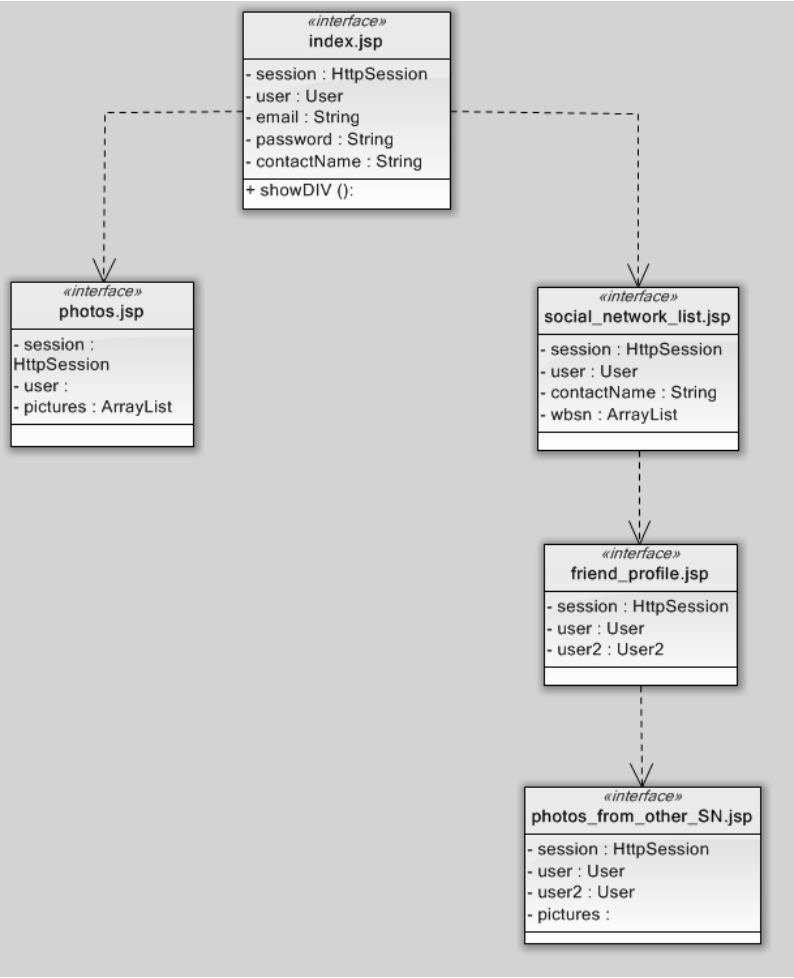

Ilustración 18: Diagrama de clases – Vista (Friendbook)

Esta red social tiene una menor cantidad de páginas que MyLeisure, debido a que unifica varios conjuntos de datos en una misma página (por ejemplo, la página de perfil incluye la lista de contactos del usuario). Las páginas que conforman esta red social son:

index.jsp:

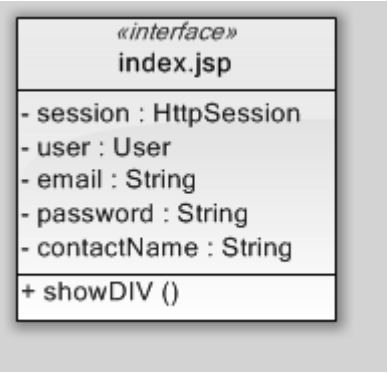

Ilustración 19: index.jsp (Friendbook)

La página *index,jsp* cuyo diagrama se observa en la Ilustración 19 contendrá en primer lugar el cuadro de "login", para que los usuarios puedan iniciar su sesión en Friendbook. Una vez exista una sesión iniciada, este cuadro desaparece, y los datos que se cargan son los correspondientes al perfil del usuario (compuesto por nombre, email, nacionalidad, colegio/universidad y edad) junto a la lista de contactos del usuario. Desde el momento en el que se insertan los datos hasta que se muestra el perfil del usuario, aparece un mensaje indicando que el sistema se encuentra "Cargando". Para mostrar este mensaje, se hace uso de la función "showDIV()".

Esta página, una vez existe una sesión iniciada, ofrece acceso tanto a las fotografías del usuario propietario de la sesión, como al perfil de alguno de sus contactos. Además, permite el cierre de sesión a través de un botón creado con este objetivo.

photos.jsp:

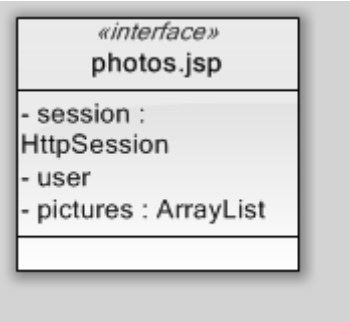

Ilustración 20: photos.jsp (Friendbook)

La interfaz *potos*.jsp (Ilustración 20) es la encargada de mostrar las fotos del usuario que ha iniciado la sesión. Las fotos aparecen de una en una en forma de columna. En el caso de sobrepasasen el tamaño de la pantalla, debe aparecer un scroll para recorrer la columna de fotos.

Permite el regreso al perfil del usuario, así como el cierre de la sesión, tal y como se comentó anteriormente.

social_network_list:

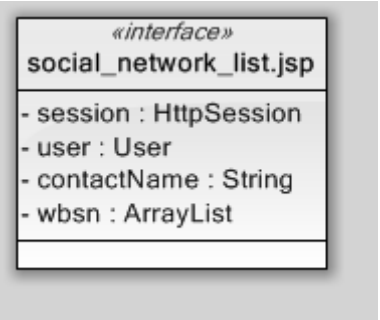

Ilustración 21: social_network_list.jsp (Friendbook)

En la Ilustración 21 se observa la página cargada en el momento en el que se seleccione a uno de los contactos del usuario que ha iniciado la sesión. Ofrece un listado de redes sociales en

las que el contacto mantiene un perfil almacenado, de manera que el usuario puede elegir de qué red social desea recuperar la información. Al hacer clic sobre una de las redes sociales mostradas, se procede a la obtención del perfil del contacto de esta red social.

friend_profile_from_other_SN.jsp:

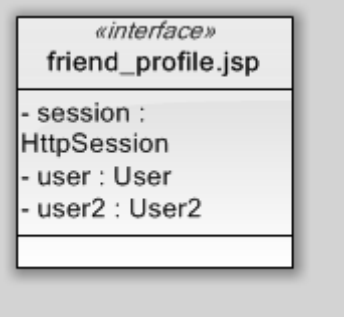

Ilustración 22: friend_profile.jsp (Friendbook)

Como muestra la Ilustración 22, el contenido de ésta página es la información de perfil del contacto que se seleccionase anteriormente. Esta información estará compuesta únicamente de sus datos personales, no aparecerá su lista de contactos. El motivo por el que ésta no aparece es que se reduce la visibilidad de perfiles a los contactos del propio usuario, y no a contactos de contactos. De esta forma, un usuario controla exactamente quién puede observar su perfil.

Junto a esta información, se ofrece acceso a las fotos del contacto visitado, así como la posibilidad de finalizar la sesión o regresar al perfil del usuario que la inició.

photos_from_other_SN.jps:

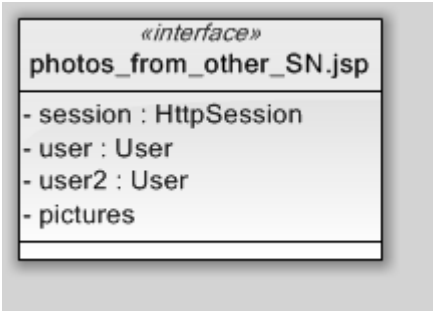

Ilustración 23: photos_from_other_SN.jsp (Friendbook)

Esta página será muy similar a *photos.jsp.* La única diferencia entre ellas es el propietario de las fotografías que se muestran, dado que en el primer caso eran las del usuario que había iniciado la sesión, y en este son las del contacto al que se ha accedido.

3.1.1.1.2 MyLeisure

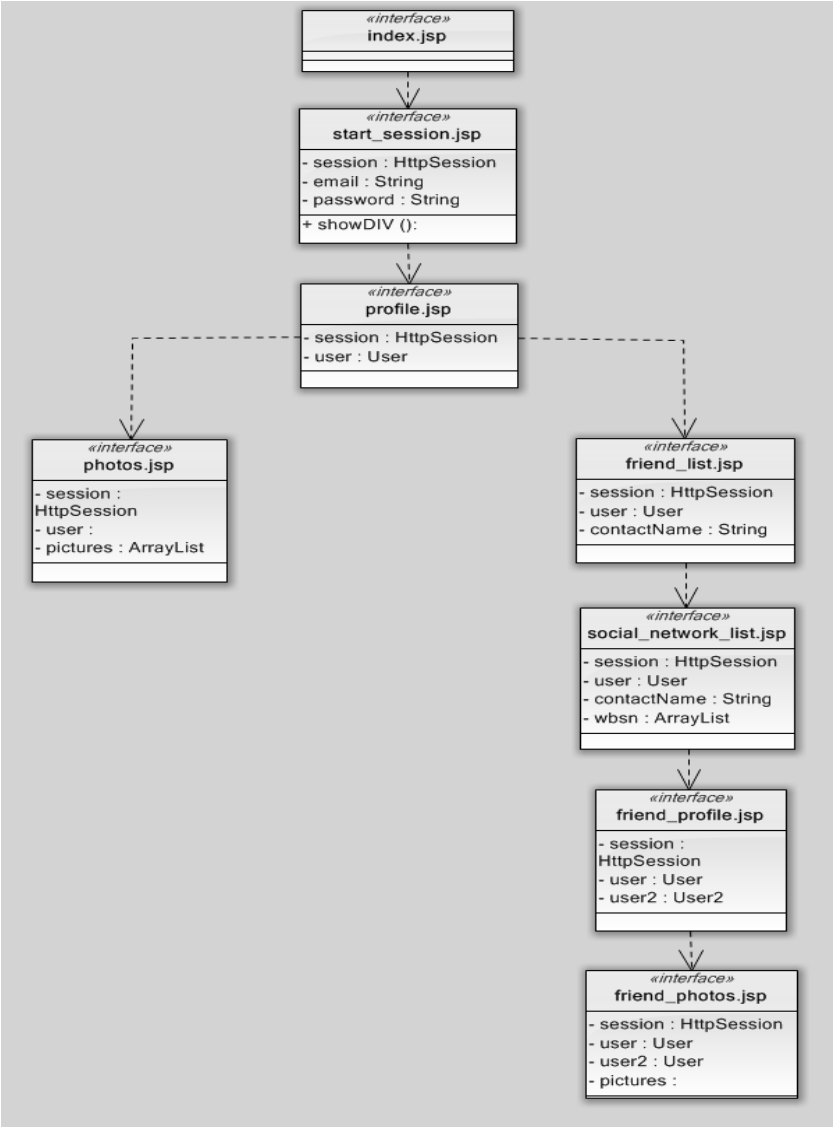

Ilustración 24: Diagrama de clases – Vista (MyLeisure)

Esta segunda red social tiene una mayor cantidad de páginas debido a, como se comentaba antes, la menor agrupación de información en una misma página, y otros detalles como la inclusión de una portada como bienvenida a la web.

A continuación se describe cada una de las páginas existentes en MyLeisure, así como sus diferencias con Friendbook:

index.jsp:

Ilustración 25: index.jsp (MyLeisure)

Al igual que en Friendbook, la primera página que muestra la red social recibe el nombre de *index.jsp*. En este caso, esta página tan solo será de bienvenida, por lo que proporciona un enlace a la página en la que se debe realizar el inicio de la sesión pero no contiene atributos ni métodos, como se puede comprobar en la Ilustración 25.

start_session.jsp:

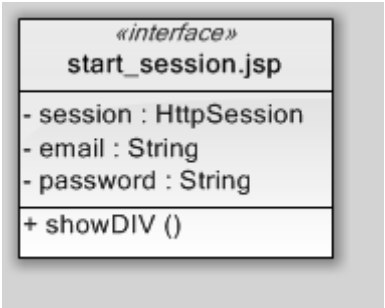

Ilustración 26: start_session.jsp (MyLeisure)

Contiene el cuadro de "login", solicitando email y contraseña para poder iniciar sesión en MyLeisure. Una vez completado el formulario de inicio de sesión, y realizado la parte correspondiente del protocolo, se cargará la página con el perfil del usuario. Durante el proceso se carga, se mostrará un mensaje indicando el estado “Cargando”, para lo cual se realiza una llamada a la función “showDIV()”.

profile.jsp:

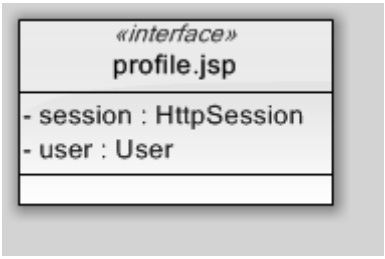

Ilustración 27: profile.jsp (MyLeisure)

Página que contendrá los datos de perfil de un usuario (nombre, email, nacionalidad, colegio/universidad, edad), tal y como se muestra en la Ilustración 27.

A parte de mostrar el perfil del contacto, ofrece acceso a sus fotografías, así como a la lista de amigos obtenida de su perfil. Por último, ofrece la posibilidad de finalizar la sesión de manera manual.

photos.jsp:

Ilustración 28: photos.jsp (MyLeisure)

Página similar a la que se ha descrito en Friendbook con el mismo nombre. Tan solo contiene las fotos del usuario que ha iniciado sesión en una lista vertical, junto a los enlaces para finalizar la sesión o regresar al perfil del usuario.

friend_list.jsp:

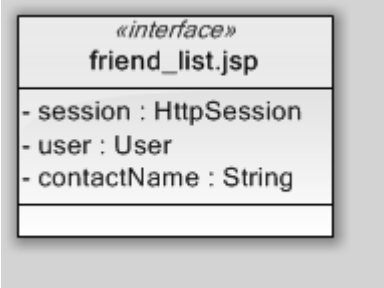

Ilustración 29: friend_list.jsp (MyLeisure)

Contiene una lista con todos los contactos del usuario que ha iniciado la sesión. Cada uno de los nombres enlazará a la página en la que se muestra el listado de redes sociales en las que se encuentra registrado este contacto. En la Ilsutración 29 se pueden observar los atributos que componen esta página.

social_network_list.jsp:

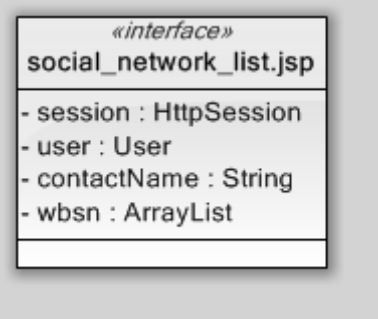

Ilustración 30: social_network_list.jsp (MyLeisure)

Listado de redes sociales en las que mantiene un perfil el contacto seleccionado. Cada una de las redes que compone este listado enlaza al correspondiente perfil del contacto.

friend_profile.jsp:

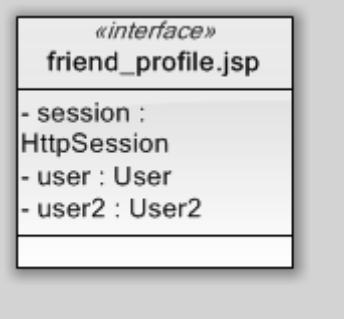

Ilustración 31: friend_profile.jsp (MyLeisure)

Página con la misma estructura y funcionalidad que *profile.jsp* (como se puede comprobar en la Ilustración 31). Las únicas diferencias entre ellas serán el objetivo, dado que una muestra el perfil del usuario que ha iniciado la sesión mientras que otra muestra la del contacto seleccionado, y el enlace a la lista de amigos, que en este caso desaparece por el motivo comentado anteriormente.

También ofrece acceso a las fotografías del contacto, así como la opción de regresar al perfil del usuario que inició la sesión, o finalizar ésta manualmente.

friend_photos.jsp:

Ilustración 32: friend_photos.jsp (MyLeisure)

Esta página es igual que las descritas anteriormente cuyo objeto es la visualización de fotografías. No aporta ninguna novedad.

3.1.1.2 Controlador

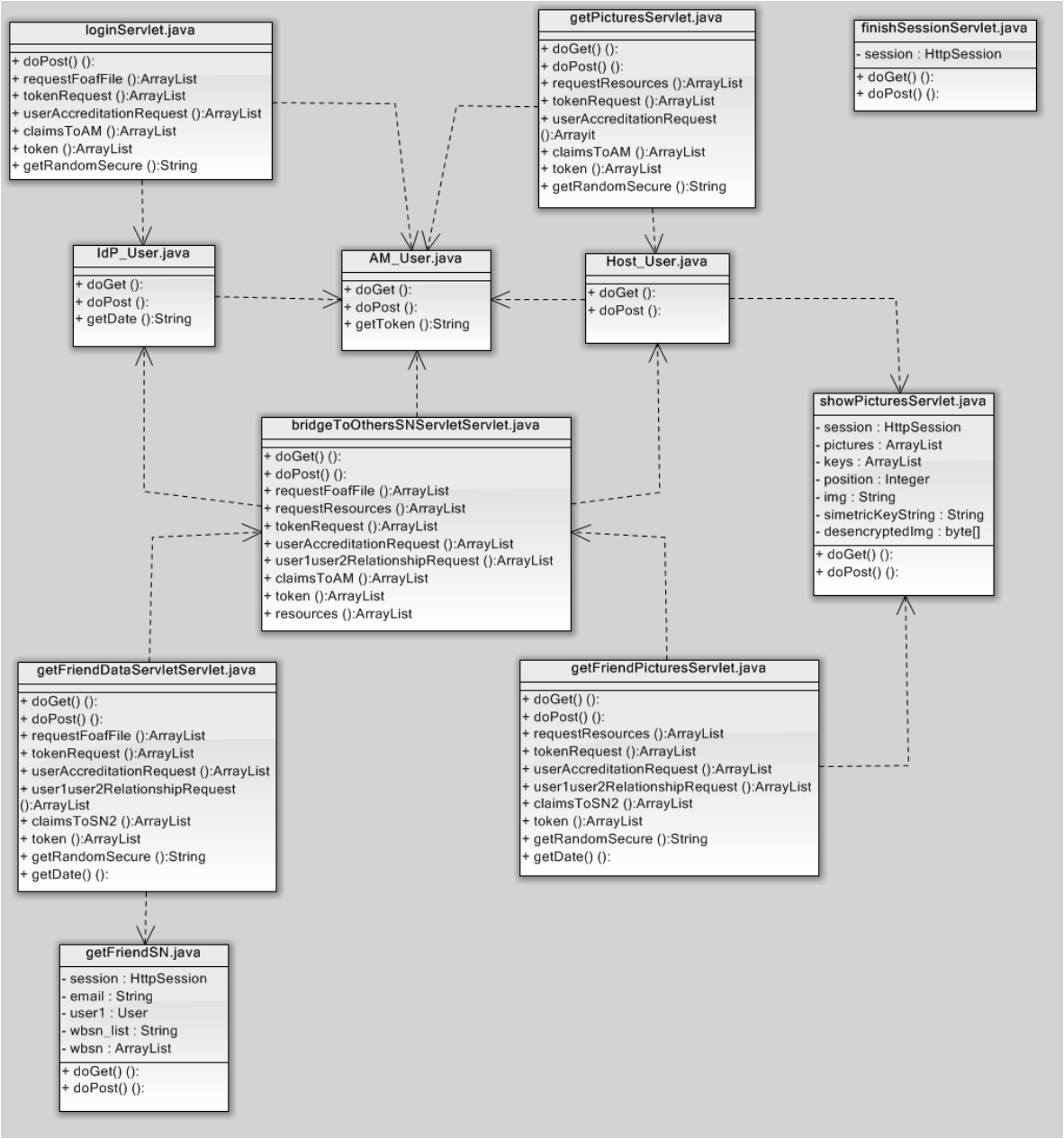

Ilustración 33: Diagrama de clases - Controlador

En esta sección se describe el elemento *Controlador*, compuesto por las clases que se observan en la Ilustración 33 y encargado de la navegabilidad entre las distintas páginas que componen la interfaz, el envío de mensajes a otras entidades, así como su recibo y procesamiento, y la gestión de datos, dado que los obtendrá de la base de datos y la ofrecerá a la interfaz que la requiera.

Tanto en las redes sociales como en el resto de servidores, el elemento controlador se implementa mediante servlets, los cuales pueden recibir parámetros y atributos desde las interfaces o desde otros servlets, así como realizar consultas a las bases de datos.

Estos servlets se caracterizan por el uso de dos métodos principales, *doGet()* y *doPost()*, los cuales ejecutarán al recibir un mensaje desde una de las entidades. A estos dos métodos

incluidos por defecto, se le añadirán los diseñados específicamente para el proyecto, que facilitarán la programación.

En ambas redes existen los mismos servlets, dado que las diferencias existentes en las interfaces no requieren de la actuación de ningún servlet específico.

Los servlets que se van a implementar para cumplir las funcionalidades de las redes sociales son:

loginServlet:

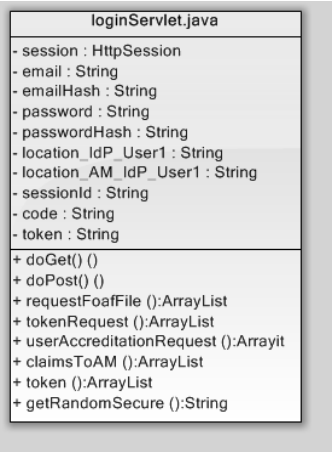

Ilustración 34: loginServlet.java

Este servlet tiene dos funciones, la de comprobar si los datos introducidos por el usuario para iniciar sesión se corresponden con los almacenados en la base de datos, y la de realizar la parte del protocolo que se encarga de obtener los datos del perfil de este usuario.

Para realizar la primera, se comenzará la ejecución a través del método *doPost()*.Este método debe llamar al método "checkLogin()", perteneciente al módulo encargado de establecer las conexiones con la base de datos, y que devolverá el resultado (verdadero o falso) de la solicitud.

Si el resultado obtenido es erróneo, se carga de nuevo la página *index.jsp*, proporcionándole un mensaje que detalle el error ocurrido.

Si el resultado fuese correcto, se procede con la segunda función del servlet, la obtención de los datos del perfil a través del protocolo.

Para cumplir la segunda funcionalidad, se debe continuar la ejecución del mencionado método *doPost()*. Este método se encarga del envío de todos los mensajes que aparecen en el protocolo cuyo objetivo es el de recuperar los datos de perfil del usuario. Además, será el encargado de recibir las respectivas respuestas y procesarlas, decidiendo cual debe ser el siguiente paso a realizar en función del tipo de mensaje que se haya recibido.

Para el proceso de envío, desde el método *doPost()* se realizan llamadas a otros métodos, que se encargan de enviar el mensaje correspondiente. Estos métodos, mencionados en la Ilustración 34, son:

- requestFoafFile(): es el encargado de enviar el mensaje con identificador "1" del protocolo para la obtención del perfil. El mensaje que genera se envía al *IdP* que contiene los datos del usuario. Los campos que incluirá este mensaje serán los definidos como fijos, es decir, el tipo de mensaje, el hash del email del usuario que ha iniciado sesión, el código y el identificador de sesión.
- tokenRequest(): envía el mensaje con identificador "3". Este mensaje se destina al *AM* asociado al IdP al que se le envió el primero. Los campos del mensaje serán de nuevo los fijos.
- userAccreditationRequest(): envía el mensaje con identificador "5" destinado al *IdP*. De nuevo el mensaje está formado únicamente por los campos fijos.
- claimsToAM(): envía el mensaje "7" del protocolo al *AM*. A los campos fijos, se unen la fecha y la firma del *IdP*, cuyos valores son los recibidos en el anterior mensaje.
- token(): es el método encargado del envío del último mensaje generado por este servlet. El identificador de dicho mensaje será el "10", y su destino el *IdP*, con el objetivo de que éste le devuelva los datos requeridos (perfil del usuario). A los campos fijos, se une el token recibido de parte del *AM*, para que el *IdP* pueda verificar la autenticación.

Junto a estos métodos encargados exclusivamente del envío de mensajes a diferentes entidades, aparecen otro nuevo método, el generador de números aleatorios (getRandomSecure()):

- getRandomSecure(): se encarga de generar un número aleatorio que será usado como identificador de sesión. Para generarlo, se hace uso de la librería "java.securiy.SecureRandom" en lugar de "java.util.Random", dado que su índice de colisiones es menor.

Una vez se finaliza el protocolo y, si todo ha funcionado correctamente, se habrá recibido el perfil del usuario. Este perfil se guardará en la sesión que se ha iniciado, de manera que en cualquier momento se pueda acceder a los datos del usuario. Tras esto, se lanza la página *index.jsp*, que es la encargada de mostrar los datos recién obtenidos.

getPicturesServlet:

Ilustración 35: getPicturesServlet.java

La función de este servlet es la de obtener las fotografías del usuario que mantiene la sesión iniciada, para lo cual deberá realizar la parte del protocolo diseñada con este objetivo.

Este proceso se realiza de manera similar al de obtención de datos de perfil, de manera que todos los métodos descritos anteriormente aparecen también en este servlet, con dos excepciones:

- requestResources(): este método realiza las mismas funciones que "requestFoafFile()". EL único cambio que se aplica es el del nombre del método, para adecuarlo al correspondiente en el protocolo.
- token(): en ésta implementación, junto a los campos fijos y el token, se enviará al *IdP* un vector de inicialización (previamente generado en el método *doPost()*) y la clave del certificado público del usuario, con el objetivo de que se cifren los recursos antes de enviarlos a la red social.

Tras recibir el último mensaje, se recuperan de éste las fotografías del usuario junto a la clave privada con la que se encontraban cifrados, y se almacenan en la sesión para posteriormente poder mostrarlos a través de la interfaz.

Para finalizar, se carga la página *photos.jsp*, para que realice los ajustes necesarios para mostrar las fotografías al usuario.

showPicturesServlet:

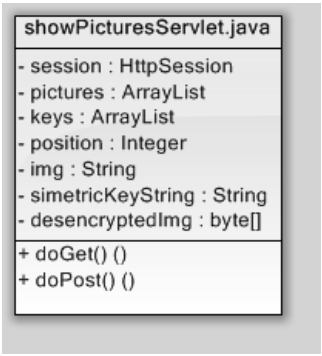

Ilustración 36: showPicturesServlet.java

Este servlet se encarga de mostrar las fotografías recibidas del *Host*, por tanto se llama desde la página *photos.jsp*. No se trata de un servlet que redirige el flujo, sino que tan solo descifra las fotografías y las "pinta" para que se muestren en la interfaz.

Para poder mostrar las fotografías, tiene que recuperarlas de la sesión, junto a las claves privadas que hay que usar para el descifrado.

Una vez el servlet está en posesión de ambas, procede a descifrar la clave privada. Para ello, habrá que utilizar la clave del certificado privado del usuario, dado que el cifrado se realiza con la pública de éste.

Cuando se finaliza el proceso y se obtiene la clave privada, tan solo habrá que usar esta para descifrar las fotografías.

Para finalizar su ejecución, el servlet indicará que el contenido que va a mostrar son imágenes ("image/jpeg"), y acto seguido muestra las fotografías.

getFriendSN:

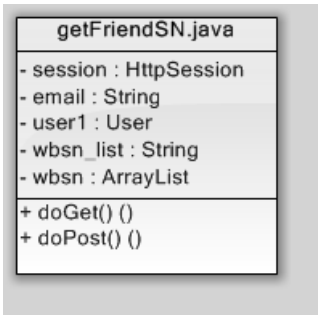

Ilustración 37: getFriendSN.java

La función de este servlet es simple: debe obtener del fichero FOAF la lista de redes sociales en las que está registrado el contacto elegido en el JSP, y guardarlo como atributo para que posteriormente sea más sencilla su recuperación.

Para realizar esta función con éxito, en primer lugar debe recuperar el hash del email del contacto al que se desea acceder. Este valor se encontrará almacenado como parámetro del *request.*

Tras esto, se recuperará el fichero FOAF del usuario y se creará un objeto "User" a partir de él. De este modo, tan solo habrá que llamar a uno de los métodos de este objeto para recuperar la lista de redes sociales.

Una vez se obtiene la lista, se almacena como atributo del request para poder recuperarla en la página hacia la que se redirigirá el flujo, *social_network_list.jsp*.

getFriendDataServlet:

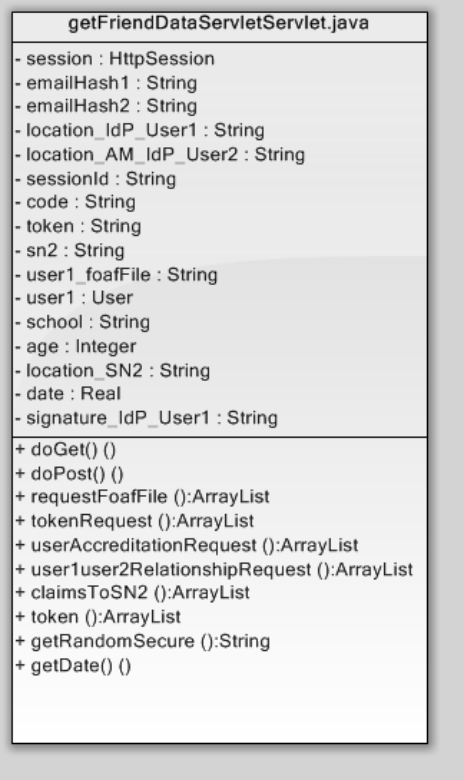

Ilustración 38: getFriendDataServlet.java

Este servlet, cuyo esquema se muestra en la Ilustración 38, es el encargado de cumplir la parte del protocolo cuyo objetivo es el de obtener los datos del perfil de uno de los contactos del usuario que ha iniciado la sesión.

Muchos de los mensajes de esta parte del protocolo son similares a los pertenecientes a la parte destinada a obtener los datos del perfil del propio usuario que inicia sesión. Este hecho conlleva que se reutilicen gran cantidad de métodos, por lo que a continuación se detallan únicamente aquellos métodos que aporten alguna novedad.

- requestFoafFile(): este método es reimplementado, dado que en esta ocasión debe mandar su firma con el objetivo de que la segunda red social pueda verificar que está estableciendo comunicación con una red social real.

La firma se aplicará sobre la cadena que resulte de la concatenación del hash del email del usuario y la fecha del sistema. Por tanto, los campos que se enviarán en el mensaje, a parte de los fijos, serán el texto firmado y la fecha del sistema en el momento de la firma.

- user1User2RelationshipRequest(): es método se encarga de enviar el mensaje con identificador ”12” a *SN2*. Con él se entrega la acreditación de que ambos usuarios han establecido una relación previamente.

  El mensaje incluye los campos fijos, junto a la firma del *IdP* del usuario que ha iniciado la sesión, y la fecha en la que se realiza esta firma.

- claimsToSN2(): envía el mensaje “17” a la entidad *SN2*. A través de este mensaje, se proporciona al *AM_IdP_User2* los requisitos necesarios para que se realice la autenticación del usuario. Junto a los campos fijos, se envía la firma del *IdP_User1* junto a la fecha en la que se ha realizado, y la edad y el centro de estudios del usuario que solicita los datos, campos con los que se comprobará si cumple las políticas de control de acceso del contacto al que intenta acceder.

Cómo se ha comentado, el resto de métodos de este servlet no se detallan por no aportar ninguna novedad. En estos métodos, tan solo se modificará el destinatario de los mensajes, que será en la mayoría de los casos la *SN2*.

getFriendPicturesServlet:

Ilustración 39: getFriendPicuresServlet

Servlet encargado de obtener los recursos (fotografías) del contacto seleccionado por el usuario. Dada la similitud de esta parte del protocolo con la dedicada a obtener el perfil del contacto, existe una gran reutilización de métodos. Por este motivo y como ha sucedido en anteriores ocasiones, tan solo se van a describir los métodos que aporten novedades respecto a los anteriores.

- *requestResources()*: el contenido de este método es el mismo que el de requestFoafFile() del servlet *getFriendDataServlet*.

La principal diferencia que existe entre este servlet y el encargado de recuperar el perfil del contacto es la información que se recibe al finalizar el protocolo. En este caso se reciben las fotografías del contacto junto a las claves que habrá que utilizar para descifrarlas. Ambos recursos se almacenan en la sesión para que posteriormente se descifren y se muestren al usuario.

Para finalizar, se realiza la llamada a *photos_from_other_SN.jsp*, que será el encargado de mostrar las imágenes recuperadas.

bridgeToOthersSNServlets:

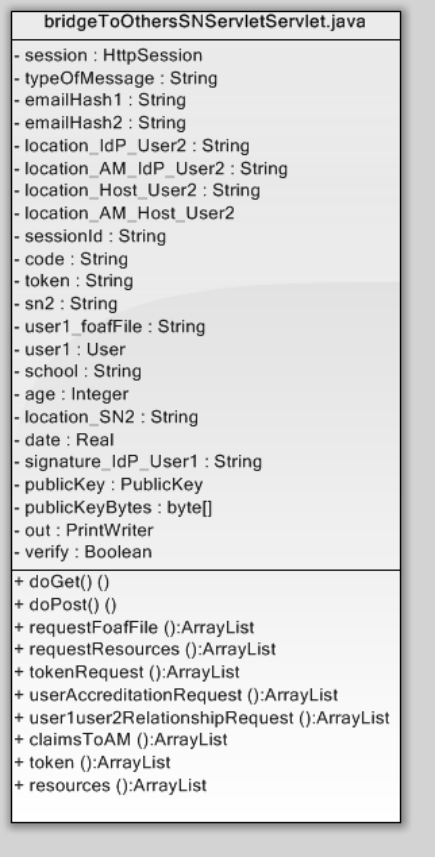

Ilustración 40: bridgeToOthersSNServlet.java

Este servlet es el encargado de recibir las solicitudes de otras redes sociales, por lo que actúa sólo cuando hay intercambios de información entre distintas redes sociales.

Principalmente se encarga de trasladar las peticiones de otra red social al destino indicado, que puede ser el *IdP*, *Host* o *AM* del usuario cuya información solicitan. Por este motivo, su operativa se basa en recoger los campos de los mensajes y generar otros iguales (reutilizando métodos) para redirigirlos.

Cabe destacar que, tras recibir el primer mensaje del protocolo desde otra red social, se lleva a cabo una verificación de la firma de esta, para comprobar que se trata de una red social aceptada y no de una suplantación.

finishSessionServlet:

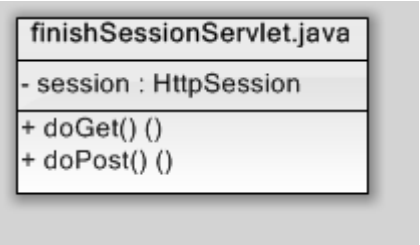

Ilustración 41: finishSessionServlet.java

La función de este servlet es la de finalizar la sesión cuando el usuario lo desee, es decir, cuando se pulse el botón dedicado al cierre de sesión manual "Cerrar sesión".

En su ejecución lleva a cabo dos operaciones. En primer lugar, invalida la sesión que se mantenía activa para el usuario, perdiendo así los datos que ésta mantenía. Acto seguido, se dirige a la aplicación a la página *index.jsp*, pidiendo un nuevo inicio de sesión.

Junto a estos servlets se hará uso de una clase JAVA llamada *User.java*. El objetivo de esta clase es definir los atributos que definen a un usuario y que se incluirán en los ficheros FOAF. Para permitir el acceso a estos atributos, incluirá los correspondientes "getters" y "setters".

La función de esta clase es la de proporcionar objetos "User", los cuales representan el perfil de un usuario. De esta forma, cuando los *IdP* proporcionen los perfiles en formato FOAF, se transformarán en estos objetos, facilitando las posteriores operaciones que se realicen con ellos.

3.1.1.3 Modelo

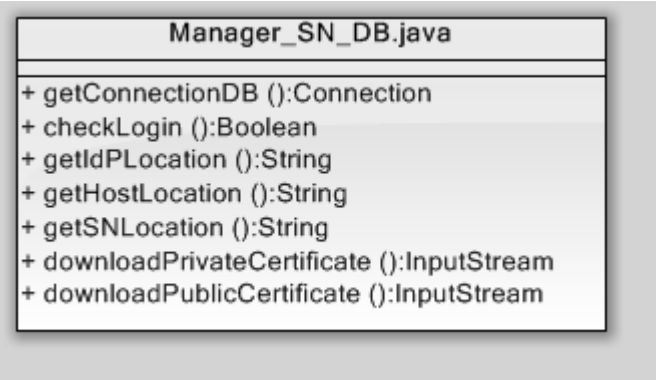

Ilustración 42: Manager_SN_DB.java

El modelo de datos es el mismo para ambas redes sociales, dado que las bases de datos contienen la misma tabla, que se describe a continuación, y usan el mismo módulo para realizar las conexiones con ésta.

Las bases de datos se han definido con los siguientes nombres: *friendbook_db* y *myleisure_db*. Ambas contienen dos tablas. La primera de ellas, usada para facilitar el login de un usuario en cada una de las redes sociales. Esta tabla contiene los siguientes campos:

- **email:** almacena el hash del email de cada uno de los usuarios registrados en la red social. Este campo es la *Primary Key* de la base de datos, dado que las búsquedas se realizan a través de él.
- **password:** hash de la contraseña elegida por el usuario para el inicio de la sesión. Cada vez que un usuario intente iniciar una sesión, se comprobará que su email y contraseña corresponden con los almacenados en ésta base de datos.
- **idp_location:** almacena la URL en la que se encuentra el servidor elegido por el usuario para almacenar sus datos de perfil. Es necesario que la red social conozca su ubicación para poder recuperar estos datos.
- **host_location**: igual que en el caso anterior, el usuario debe facilitar también la ubicación del servidor donde se alojan sus fotografías, para que la red social pueda recuperarlas haciendo uso del protocolo.

La segunda de las tablas está destinada, por un lado, a poder verificar las firmas de las redes sociales que intenten contactar con ellas y, por otro, a almacenar la dirección de las redes sociales con las que puede conectar, ligadas a sus correspondientes nombres. De esta manera, cuando un usuario desee obtener información de un contacto de otra red social, se obtendrá el nombre de la red social a partir de la lista de estas, y gracias a él se conseguirá la localización en la base de datos.

Conociendo los dos propósitos de esta tabla, se determina que debe contener los tres campos mencionados: nombre identificativo de cada red social, localización que se debe usar para contactar con ella, y certificado público de ésta para verificar las posibles firmas recibidas.

Para facilitar las llamadas a las bases de datos, se va a construir un módulo encargado de establecer las conexiones con éstas, así como recuperar, insertar o actualizar sus valores.

Este módulo (Ilustración 42) recibe el nombre de *Manager_SN_DB*, es utilizado tanto por *Friendbook* como por *MyLeisure*, y está compuesto de los siguientes métodos:

- **getConnectionDB:** este método recibe como parámetro el nombre de la base de datos con la que se desea establecer conexión (parámetro usado para distinguir *Friendbook* de *MyLeisure*). Se encarga de crear una conexión con la base de datos indicada, y devolver el objeto "Connection" que se usará para realizar las posteriores consultas.
- **checkLogin():** este método se encarga de comprobar si los datos introducidos por el usuario son correctos y se puede comenzar el protocolo para obtener su perfil. Para ello, recibe tres parámetros. En primer lugar la base de datos donde debe verificar estos datos (*Friendbook* o *MyLeisure*). Como segundo y tercer parámetro recibe los resúmenes del email y de la contraseña, con los que va a realizar la búsqueda. Si la consulta devuelve resultados, se retornará el valor "true", y en caso contrario se retornará "false".
- **getIdPLocation():** su función es la de recuperar la dirección del *IdP* en el que se encuentra el perfil del usuario. De nuevo se recibe la base de datos en la que se debe realizar la consulta, junto al hash del email, que identificará al usuario cuyo perfil se desea recuperar.
- **getHostLocation():** método similar al descrito previamente. Recibe los mismos parámetros y realiza la misma consulta, devolviendo en este caso la dirección del *Host* donde el usuario almacena sus fotografías, en lugar de la del *IdP*.
- **getSNLocation():** el fin de este método es el mismo que el de los dos anteriores, recuperar la localización de otro servidor con el que se desea conectar. En este caso, su busca la dirección de otra red social, de la que se desea recuperar información de alguno de los contactos del usuario que inició sesión en la propia red social. En este caso, en lugar de recibir el hash del email del usuario, se recibirá el identificador de la red social cuya localización se desea recuperar. Como se ha comentado anteriormente, este identificador será el nombre de la red social.

  En esta implementación en la que tan solo existen dos redes sociales, solo existirá el registro de la red social contrario. Según se fuesen incorporando más redes sociales, aumentará el número de registros, momento en el cual el identificador cobrará su verdadera importancia.
- **downloadPrivateCertificate():** cada red social debe almacenar su certificado privado con el fin de poder firmar el primer mensaje que mande a otra red social (método usado para que la segunda red social identifique a la primera). Este certificado se va a almacenar en una tabla específica para él, con nombre "my_cert", y un único campo donde se almacenará el certificado.

  Este método se encarga únicamente de recuperar este certificado y devolverlo como un "InputStream" para su posterior uso.
- **downloadPublicCertificate():** en el momento en que una red social recibe un primer mensaje desde otra distinta, debe comprobar la firma que se incluye en este mensaje. Para ello, es necesario disponer del certificado público de esta segunda red social, el cual se encuentra en la base de datos.

  Este método se encarga de recuperar este certificado, realizando una consulta a través del identificador de la red social, de igual manera que se realizó en "getSNLocation()".

El certificado obtenido se devolverá como un "InputStream", con el cual se podrá realizar la verificación de la firma.

3.1.2 Identity Provider (IdP)

Esta sección describe el diseño de los servidores que alojan y proporcionan, cuando se les requiere, la información correspondiente al perfil de cada uno de los usuarios de las redes sociales.

Como ya se ha mencionado en este documento, podrán existir múltiples *IdPs*, de manera que cada usuario almacene sus datos en uno distinto. Aun así, todos deben mantener la misma estructura y cumplir el protocolo.

A continuación se va a detallar la estructura de estos servidores y las funciones que realizan. Esta descripción se divide en dos elementos, el controlador y el modelo de datos.

3.1.2.1 Controlador

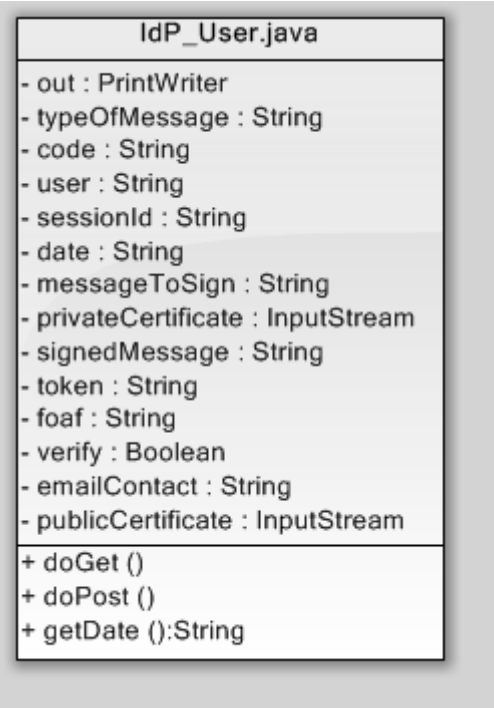

Ilustración 43: Controlador IdP_User.java

De la misma manera que sucede en las redes sociales, y sucederá en el resto de servidores, el elemento controlador se compone de servlets. En este caso, tan solo un servlet, *IdP_UserX* (donde la "X" representa el número de usuario que contiene sus datos en el servidor), se encarga de cumplir el protocolo.

La función de los *IdP* en el protocolo es la de contestar las peticiones de las redes sociales, las cuales solicitan los perfiles de los usuarios. A cada una de estas peticiones se debe contestar

en función de los datos recibidos, no pudiendo facilitar dichos perfiles hasta el momento en que se haya realizado la autenticación correctamente. El elemento que determina si se ha realizado la autenticación es el "token" generado por los *AM*.

Dado que su función es responder solicitudes, la primera acción que debe llevar a cabo *IdP_UserX* es la de extraer los campos del mensaje de solicitud que recibe, en concreto el que indica el tipo de mensaje recibido. En función de éste, se determina cuáles son las operaciones que se deben realizar, las cuales deben finalizar con una respuesta a la entidad que realice la solicitud.

A continuación se van a describir las acciones que se deben llevar a cabo dependiendo del valor del campo 'typeOfMessage', clasificadas por este mismo valor:

- **requestFoafFile:** este tipo de mensaje indica que se solicita el perfil de un usuario. Dado que no se aporta ningún tipo de autenticación, se debe responder requiriendo el "token" que verifique la autenticación del usuario.

  Para que la red social pueda obtener dicho "token", se debe proporcionar la localización del *AM* que se la puede proporcionar. Este campo ("location_AM_IdP") es el único variable que incluirá el mensaje de respuesta a la solicitud.

- **userAccreditationRequest:** en esta ocasión la red social solicita que se firme la acreditación del usuario, para poder verificar ante el *AM* la autenticidad del usuario.

  Dicha firma se debe realizar con el certificado privado perteneciente al proveedor de identidad.

  Como en todos los mensajes en los que se incluye una firma, a parte del campo "signedMessage" se debe incluir la fecha en la que se realiza esta firma.

- **token:** en este mensaje, la red social proporciona el "token" solicitado como prueba de la autenticidad del usuario.

  Antes de facilitar la información a la red social, se debe contrastar con el *AM* que el token es válido. Para ello, se ha de enviar un mensaje a esta entidad con el "token" recibido. De esta solicitud se recibirá una respuesta, cuyo tipo de mensaje será "tokenValidation".

- **tokenValidation:** este mensaje, al contrario del resto ya descritos, se recibe de parte del *AM*. Debe indicar si el token proporcionado por la red social es válido o no.

  En el primer caso, se debe enviar a la red social el perfil del usuario solicitado, en formato FOAF. Este campo es el único de los variables que incluye este mensaje.

  En el caso de que la entidad *AM* determine que el "token" no es válido, se debe devolver un mensaje de error indicando este hecho.

  De esta forma, el *IdP* concederá los datos del perfil del usuario que ha iniciado sesión en el sistema. Cuando los datos solicitados no son los del propio usuario sino los de uno de sus contactos, se incluye un nuevo paso para el *IdP*:

- **user1user2RelationshipRequest:** en este mensaje se recibe la acreditación de que el usuario del que se solicitan los datos y el solicitante mantienen algún tipo de relación. Esta acreditación debe estar firmada por el *IdP* del usuario que solicita los datos.

  Ante este tipo de mensaje, la primera acción a llevar a cabo debe ser la verificación de la firma. Si esta resultase correcta, se debe devolver el mensaje a la red social, para

que esta pueda presentarlo ante el *AM* (una vez verificada la acreditación, la red social podrá saltarse este paso en sucesivas solicitudes).

En el caso de que no fuese correcta, se contesta con un mensaje que indique el error que se ha producido.

Finalizada la descripción de este controlador, mostrado en la Ilustración 43, se procede a la descripción del modelo de datos.

3.1.2.2 Modelo

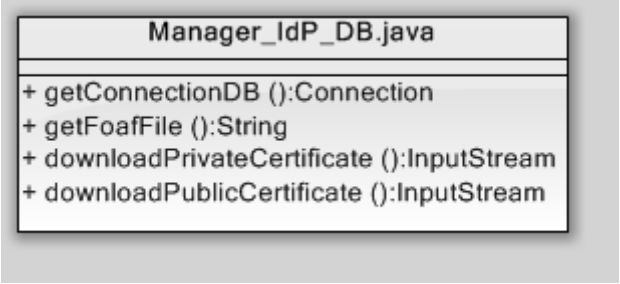

Ilustración 44: Manager_IdP_DB.java

Cada entidad *IdP* está ligada a una base de datos, identificada mediante el nombre "IdP_DB". El bien más preciado que almacenan estas bases de datos es el perfil de los usuarios, más concretamente, el fichero FOAF que lo define. Para almacenar estos ficheros existe una tabla (foaf_files), en la que se encontrará cada perfil enlazado al hash del usuario propietario. Por tanto, estos dos campos serán los que compongan esta tabla, siendo el hash del email la *Primary Key*.

Por otro lado, debe existir una segunda tabla encargada de almacenar certificados. Por un lado el propio certificado privado del *IdP* para poder realizar las firmas oportunas. Por otro, los certificados públicos del resto de *IdP* para poder verificar las firmas que reciba.

Para acceder a ambas tablas y realizar las consultas oportunas, se dispone de un módulo JAVA, que aporta métodos para cada una de las funciones necesarias. Entre estos métodos, algunos realizan las mismas funciones descritas en la Sección 3.1.1.3, por los que no se describirán de nuevo:

- **getConnection:** método descrito en la Sección 3.1.1.3.
- **getFoafFile:** este método es el encargado de recuperar de la base de datos el perfil de un usuario. Para ello, recibe dos parámetros: por un lado, el nombre de la base de datos de la que debe recuperar dicho perfil (*IdP_User1_DB* o *IdP_User2_DB*); por otro lado recibe el hash del email del usuario cuyo perfil se debe recuperar. El método retorna el fichero FOAF que contiene el perfil. En lugar de devolver como fichero, devolverá su serialización, por lo que aparecerá como una cadena de texto.

- **downloadPrivateCertificate():** método descrito en la Sección 3.1.1.3. La única diferencia será el nombre de la base de datos que se recibe por parámetro, que en este caso podrá ser *IdP_User1_DB* o *IdP_User2_DB.*
- **downloadPublicCertificate():** método descrito en la Sección 3.1.1.3. De nuevo la diferencia reside en el nombre de base de datos que se recibe por parámetro.

3.1.3 Host

El servidor encargado de la administración de las fotografías (*Host*), es similar en la mayoría de los aspectos al recién detallado *IdP*. La principal diferencia es que el *Host* no tiene que realizar firmas, ya que no se encarga de proporcionar ningún tipo de autenticación.

Del mismo modo que se ha realizado en las anteriores secciones, el diseño del *Host* se va a dividir en dos elementos, el *Controlador* y el *Modelo*.

3.1.3.1 Controlador

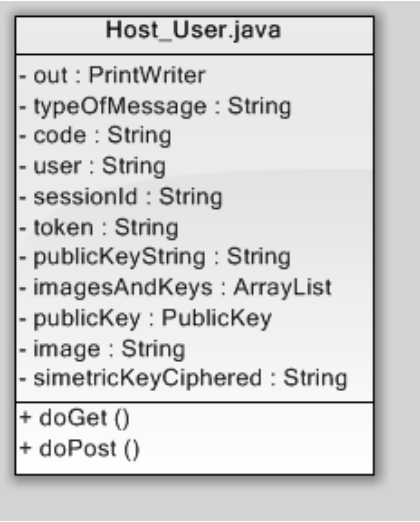

Ilustración 45: Controlador Host_User.java

En la versión más simple del protocolo, es decir, el acceso a fotografías del propio usuario que ha iniciado la sesión, el controlador del *Host* es bastante sencillo, ya que tan solo va a recibir dos tipos de solicitudes:

- **requestResources():** se trata de un mensaje para solicitar las fotografías del usuario. El mensaje es similar al "requestFoafFile", y la respuesta que se debe proporcionar también lo es. El mensaje de respuesta contiene los campos fijos junto a la dirección del *AM_Host* con el que el usuario debe negociar la autenticación.
- **token:** el proceso a realizar al recibir este tipo de mensaje se puede dividir en dos partes. En primer lugar, se comprueba que el token recibido es válido, para lo cual

debe enviarse al *AM* encargado de generarlos. Esta parte del proceso se hace igual que en la obtención de perfiles.

En la segunda parte es en la que se encuentran las diferencias. En esta parte se procede al envío de las fotografías, para lo cual previamente se debe cifrar la clave con la que se procederá al descifrado total de la imagen al final del protocolo.

Como se comentó en la Sección 2.1.2.3, el cifrado se realiza mediante el algoritmo AES, y utilizando la clave pública del usuario.

Cuando el objetivo del *Host* es el de proporcionar las imágenes de un contacto del usuario que mantiene la sesión, esta entidad recibe un tipo de mensaje más, el "user1user2RelationshipRequest".

La actuación del *Host* al recibir este tipo de mensaje es la misma que la que lleva a cabo el *IdP*, hecho por el que no se detallará de nuevo (explicación en la Sección 3.1.2.1).

3.1.3.2 Modelo

Ilustración 46: Manager_Host_DB.java

El modelo de datos del *Host* también es bastante parecido al empleado en el *IdP*. La base de datos *Host_DB* contiene dos tablas. La primera tabla contiene las fotografías de los usuarios. Dado que éstas se almacenan cifradas, las tablas deben contener junto a cada fotografía la clave privada que se debe usar para descifrarla una vez esta se encuentre en la propia red social.

Estos dos campos están acompañados por el hash del email del propietario de las fotografías. Este hash, al igual que sucedía en el resto de las bases de datos, es la clave primaria de la tabla, y será el usado para poder obtener las fotografías del usuario deseado únicamente.

Por otro lado, debe existir una tabla para el almacenamiento de certificados, ya que el *Host* debe poseer los certificados públicos de aquellas entidades de las que tenga que verificar su firma (*IdP*). Además de estos certificados, debe almacenar los públicos de los usuarios que desean recibir alguna fotografía, para que con ellos pueda cifrarlas de manera previa al envío.

Para el acceso a la base de datos, se utiliza un módulo JAVA cuyos métodos tienen la misma funcionalidad que los descritos en la Sección 3.1.2.2, con una excepción:

- **getImagesAndKeys():** método encargado de descargar las fotografías de uno de los usuarios junto con las claves con las que habrá que descifrarlas. Para ello recibe dos parámetros, la base de datos en la que deben realizar la consulta, y el hash del email

del usuario del que se desean descargar las fotos. Realizará una consulta en la base de datos para extraer todas las fotografías asociadas al correspondiente hash, e irá insertando en un ArrayList cada imagen con su clave simétrica. Para finalizar, retornará este ArrayList.

3.1.4 Authentication Manager (AM)

Para finalizar el diseño de cada una de las entidades, se procede con la entidad encargada de verificar las políticas de control de acceso, los *AM*.

Como se ha comentado en el análisis del sistema, existen dos tipos de *AM*, los que actúan en la solicitud de perfiles y se encuentran asociados a los *IdP*, y los que lo hacen cuando se requieren los recursos, que están asociados a los *Host*.

A pesar de ser dos tipos de *AM* diferentes, ambos reciben los mismos tipos de mensajes y generan respuestas muy similares, por tanto tan solo se realiza el diseño de uno de ellos, siendo aplicable a los dos.

Este hecho es el que conlleva que, como se ha comentado anteriormente, pudiese realizarse una futura modificación en la que un solo *AM* actuase en ambas partes del protocolo.

El diseño de esta entidad se va a dividir de la misma forma que se hizo con las anteriores. En primer lugar se detalla el controlador de la entidad y, posteriormente, el modelo de datos usado.

3.1.4.1 Controlador

Ilustración 47: Controlador AM_User.java

La parte que controla el funcionamiento de los *AM* se implementa, como en anteriores ocasiones, mediante un servlet. Este servlet, presentado en la Ilustración 47, recibe distintos nombres en función de si está dedicado a la autenticación para *IdP* o para *Host*. En cualquier caso, siempre se define mediante sus siglas identificadoras "AM", seguido del tipo de

servidor con el que colabora, y del usuario al que identifica (por ejemplo: *AM_IdP_User1*, *AM_Host_User2*).

El funcionamiento de este servidor se basa en la recepción de solicitudes identificadas por un tipo de mensaje, la toma de decisiones acerca de estas y la posterior respuesta. Por lo tanto, su funcionamiento es similar al de los *IdP* y los *Host*, diferenciándoles su objetivo.

En este caso, los distintos tipos de solicitudes que se van a recibir y las operaciones que se van a llevar a cabo son:

- **tokenRequest:** con este mensaje, la red social solicita al *AM* el "token" que *IdP* o *Host* le ha solicitado.

  Dado que la red social no aporta una acreditación del usuario firmada por el *IdP*, el *AM* debe solicitársela para que cuando la red social la obtenga, se la facilite en un posterior mensaje, y de esta manera se pueda realizar la autenticación correctamente y proporcionar el "token".

- **claimsToAM:** en este mensaje, la red social debe facilitar la acreditación que se le requería en la respuesta de la solicitud anterior.

  Antes de generar el "token", se debe comprobar que la firma de la acreditación es correcta. Por tanto, se debe verificar la firma haciendo uso del certificado público del *IdP* o el *Host*, el cual se encuentra en la base de datos del *AM*.

  Si la verificación no fuese correcta, se le indica este resultado a la red social y se finaliza el proceso, ya que de este modo no se puede completar la autenticación.

  Si por el contrario la verificación fuese correcta, se ha de generar un token y proporcionárselo a la red social, junto al tipo de token del que se trata y el tiempo de validez de éste.

Este servlet tiene una funcionalidad actual cuando la información que se desea obtener no es la del propio usuario sino la de un contacto. Esta funcionalidad es la de verificar las que se cumplen las políticas de control de acceso. Para ello, debe comprobar que la edad proporcionada por la red social se encuentra en el rango aceptado por el usuario, así como que pertenece al centro de estudios indicado en estas políticas.

Si ambas condiciones se cumplen, se continúa con la generación y entrega del "token". En caso contrario, se debe informar a la red social de esta situación, evitando así que se proporcione la información solicitada.

- **tokenValidation:** este tipo de mensaje ha sido detallado en las secciones 3.1.2.1 y 3.1.3.1, dado que se intercambia con *IdP* y *Host*. En este mensaje se recibe el "token" que la red social que busca la autenticación ha proporcionado a uno de estos dos servidores. Para que la autenticación sea correcta, este "token" debe ser el generado por el *AM* y proporcionado a la red social cuando se recibe un mensaje de tipo "claimsToAM". Para comprobarlo, se ha de descargar de la base de datos el token que se ha otorgado a la red social y compararlos. Acto seguido, se proporciona a la entidad emisora la respuesta obtenida, para que pueda finalizarse la autenticación o, finalmente, denegarse.

3.1.4.2 Modelo

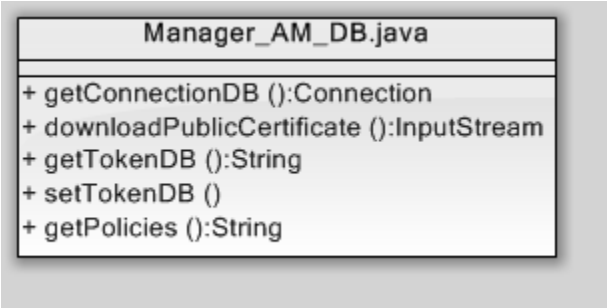

Ilustración 48: Manager_AM_DB.java

El servidor de autenticación contiene una base de datos en al que almacenar los certificados públicos con los que verificar las firmas de *IdP*s, los "token" durante el período de tiempo en el que estos sean válidos, y las políticas de control de acceso de los usuarios.

El nombre de estas bases de datos está formado por las siglas que lo identifican como autenticador, las referentes al servidor con el que colaboran, y las siglas "DB (Database)" para indicar que se trata de una base de datos. De esta forma, un ejemplo de nombre sería *AM_Host_DB*.

Las bases de datos de los *AM* están formadas por tres tablas:

- **IdPs_Certs:** esta tabla está destinada al almacenaje de certificados, más concretamente los certificados públicos de los distintos *IdPs* con los que se interactúe. Esta tabla se compone de dos campos, uno guarda los certificados ya comentados, y el otro usado para identificar el propietario de este certificado. Dado que los propietarios son otros servidores, el identificador será la URL de localización de éstos.
- **Tokens:** por otro lado se tienen que almacenar los "token" generados y proporcionados a las redes sociales. Estos "tokens" tan solo se van a almacenar durante su período de validez, por lo que transcurridas las dos horas, se borrarán de la base de datos. En esta tabla, junto a los "tokens" se encuentran los hash de los emails de los usuarios a los que se les concede, campo que será clave primaria.
- **Policies:** por último se han de almacenar las políticas de control de acceso de los usuarios. Al igual que los token se identificarán mediante el hash del email del usuario propietario. Junto a estos, aparecen otros dos campos. En primer lugar el que indica el rango de edad al que se le permite acceder a los datos. Por otro lado, el centro de estudios al que se debe pertenecer para obtener esta información.

Para facilitar la conexión y el acceso a la base de datos, se diseña un módulo JAVA. Este módulo lo componen los siguientes métodos:

- **downloadPublicCertificate():** método descrito en la Sección 3.1.1.3. Las bases de datos a las que puede acceder este método en este caso son las correspondientes a los *AM* de la aplicación, tanto los dedicados a los *IdPs* como a los *Host*.
- **getTokenDB():** este método se encarga de obtener el token que se ha proporcionado a un usuario, con el fin de verificar que el token que ha proporcionado dicho usuario al *IdP* o al *Host* es válido. Para obtener dicho token, se realiza una consulta buscando a

través del hash del email del usuario. Ésta búsqueda solo proporciona un resultado, dado que cada vez que se genera un token, el anterior se elimina de la base de datos.

- **getPolicies():** método similar al descrito anteriormente. En esta ocasión, se busca obtener las políticas de acceso de un usuario, por lo que la búsqueda se realizará de nuevo mediante el hash del email de éste. Cuando se modifican las políticas, las anteriores se eliminan, por lo que solo existe un resultado posible.

3.1.5 Otros módulos del sistema

Junto al desarrollo de las entidades participantes en la aplicación, se van a desarrollar otros módulos JAVA, que colaborarán con una o varias de estas entidades y simplificarán su funcionamiento.

Entre estos módulos se encuentran los dedicados a gestionar las bases de datos (ya comentados en las secciones 3.1.1.3, 3.1.2.2 y 3.1.4.2, por lo que no se describen a continuación), el encargado de realizar los procesos de cifrado y firma, y el encargado de crear registros (logs) del funcionamiento de la aplicación.

A continuación se describe el funcionamiento de estos módulos, detallando cuál o cuáles de las entidades hacen uso de ellos.

3.1.5.1 Módulo de firma, cifrado y funciones resumen

Ilustración 49: Crypt.java

Este módulo está formado por la clase *Crypt.java* (Ilustración 49), la cual se añade a cada uno de los servidores que necesiten hacer uso de alguna de sus funciones. En esta clase en encuentran todos los métodos necesarios para realizar las firmas y verificarlas, cifrar y descifrar datos (principalmente los recursos de los usuarios), y realizar las funciones resumen de la información que se desee.

En primer lugar, dado que serán los primeros métodos de los que se hará uso, se van a describir las funciones referentes a la creación de hash o resúmenes:

- **gethashSHA1():** este método es el principal encargado de realizar la función resumen de cualquier cadena de texto que se recibiese como parámetro. Para ello, emplea, como su nombre indica, el algoritmo SHA-1, junto a la librería "security", o más concretamente, "security.MessageDigest". El resultado de aplicar la función resumen es un array de bytes, por lo que para poder devolver un String que posteriormente sea más cómodo de manejar, se realiza una llamada al método "convertBytesToHexadecimalString()", el cual se detalla a continuación.
- **convertBytesToHexadecimalString():** este segundo método recibe como parámetro el array de bytes resultado de aplicar el algoritmo SHA-1 a una cadena de texto. La función en este caso recorrerá dicho array, convirtiendo cada byte a hexadecimal y posteriormente concatenándolo a una cadena de texto. Una vez recorrido todo el array, la cadena resultante será el valor de retorno de dicho método y, a su vez, el retorno del método descrito anteriormente.

Ambas funciones son siempre usadas de manera conjunta, puesta que la primera hace uso de la segunda para poder devolver su resultado correctamente. Su uso será frecuente, ya que se utilizan en el momento del login, para obtener el resumen tanto de la dirección de correo electrónico como de la contraseña, así como cada vez que se desea recuperar información de un contacto, ya que será el resumen de sus emails el campo identificativo para poder extraer ésta información de las bases de datos.

En segundo lugar, se describen los métodos necesarios para realizar las firmas utilizadas en la aplicación, y su posterior verificación. Para simplificar su uso desde cada uno de los servidores, se han creado dos métodos, uno encargado de firmar y otro de verificar la firma:

- **sign():** los parámetros que recibe son el certificado privado con el que se va a realizar la firma como "InputStream", la cadena que forma el texto a firmar, la clave del certificado privado mencionado y el algoritmo que se desea utilizar.

  Para realizar la firma, se extrae, haciendo uso del certificado privado y su contraseña, la clave privada con la que se firmará el texto. A continuación se realiza la firma mediante el algoritmo seleccionado y recibido como parámetro, obteniendo el resultado como array de bytes. De igual manera que se hizo en anteriores ocasiones, se transforma este array en una cadena de texto (String), siendo este último valor el que retornará el método.

- **verify_sign():** en esta segunda función, los parámetros recibidos son: certificado público que debe verificar la firma (en formato "InputStream"), el texto firmado y el texto original previo a la firma, y el algoritmo que se usará para la verificación.

  El proceso es similar al anterior. En primer lugar se obtiene la clave del certificado, en este caso, la clave pública. Se realiza la firma sobre el texto original, y si al comparar el resultado con el texto firmado con la clave privada se comprueba que ambos son iguales, se determina que la firma se encuentra en condiciones y el mensaje no ha sido modificado. En este caso, el método retornaría el valor "true", y en caso contrario, se devolvería "false".

El uso de cada una de estas funciones está indicado en la Sección 2.1.1.3, en la que se explica el protocolo implementado.

Para finalizar la explicación de este módulo, se procede a describir las funciones necesarias para realizar el cifrado y descifrado con clave asimétrica, usado sobre la clave simétrica almacenada en base de datos junto a cada una de las imágenes.

Además, se va a describir el método de descifrado de imágenes haciendo uso de la comentada clave simétrica y el algoritmo AES.

El diseño de estos cifrados posee ciertas similitudes con el patrón Facade, dado que desde la interfaz (fachada en el patrón) se seleccionará el algoritmo de cifrado a utilizar (por defecto RSA y AES respectivamente), y sin realizar ningún otro cambio, el módulo criptográfico aplicará el cifrado según el algoritmo seleccionado.

Esta descripción comenzará por los métodos referentes al cifrado de clave asimétrica:

- **generateKey():** método encargado de generar un par de claves para realizar el cifrado y descifrado. Debe recibir como parámetros el algoritmo para el que se desean las claves, así como su tamaño (si el tamaño no fuese válido para el algoritmo, se produciría un error).
- **encrypt():** realiza el cifrado de clave asimétrica sobre el texto plano que se recibe como parámetro y haciendo uso de la clave pública que se recibe del mismo modo. Cabe destacar que el tercer parámetro que recibe este método debe indicar el algoritmo a usar, el modo de cifrado y el tipo de padding.
- **decrypt():** realiza el descifrado del texto cifrado que recibe como parámetro haciendo uso de una clave privada. De igual manera que para el cifrado, se debe proporcionar el algoritmo de cifrado, el modo y el tipo de padding.

Conociendo los métodos utilizados para cifrar y descifrar haciendo uso de claves públicas o simétricas respectivamente, se presenta el método encargado de descifrar las fotografías descargadas de la base de datos, haciendo uso de la clave simétrica que se descarga junto a ellas:

- **decrypt_AES():** realiza el descifrado de imágenes, para lo que debe recibir por parámetros una imagen y la correspondiente clave simétrica con la que se cifró. En este método se da por hecho que el algoritmo utilizado es AES, dado que es el usado en el cifrado.

Cabe mencionar que se han incluido dos métodos “convertToArrayBytes()” y “convertToString()”, para facilitar las transformaciones entre ambos tipos de datos en los momentos de cifrado, descifrado, y envío de claves e imágenes.

3.1.5.2 Módulo de registro de mensajes e incidencias

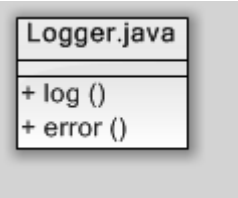

Ilustración 50: Logger.java

Este módulo se implementa en la clase *Logger.java* (Ilustración 50), la cual va a ser usada desde todas las entidades que componen esta aplicación.

La función principal de este módulo es la de registrar cada uno de los mensajes que se intercambian las entidades de la aplicación como parte del protocolo. De ésta forma se puede verificar que el protocolo se lleva a cabo correctamente, y en caso de que surgiese algún error en las conexiones se podría detectar de una manera más fácil.

Con este fin se desarrolla un método llamado "log()". Este método recibe dos parámetros, un número entero y un mensaje. El número entero debe indicar la red social principal en el protocolo, es decir, aquella en la que el usuario ha iniciado la sesión. Dado que este proyecto se compone de dos redes sociales, el número "1" identificará a *Friendbook* mientras que el "2" a *MyLeisure*. El segundo parámetro trata una parte del mensaje que se va a registrar en el fichero, concretamente la que indica qué servlet genera el mensaje y cuál es su destino, así como qué campos forman dicho mensaje.

Conocidos los parámetros del método, se procede a detallar su funcionamiento. Para comenzar, cada vez que se realice una llamada a este método, se obtiene la fecha del sistema con el formato "día-mes-año". Esta fecha es utilizada para crear, en caso de que no existiese previamente, un fichero en el que se guarde cada uno de los mensajes. Si el fichero existiese, los mensajes se incluirán al final de éste. Dado que el formato incluye únicamente el día, el mes y el año, se entiende que se va a generar un fichero de registro diferente cada día.

Una vez se apunta al fichero en el que se va a registrar el mensaje, se procede a la generación de éste. Para ello se obtiene de nuevo la fecha del sistema, esta vez incluyendo la hora exacta a la que se genera el mensaje, con el fin de llevar un control absoluto del sistema. Este nuevo formato de fecha da comienzo al mensaje a almacenar, el cual se completa con el mensaje recibido mediante parámetro incluido entre paréntesis, obteniendo un mensaje como éste:

```
Message at 19-08-2012 10:59:16(from=loginServlet, to=IdP_User1,typeOfMessage=requestFoafFile,
        code=XYZ, user=85136c79cbf9fe36bb9d05d0639c70c265c18d37, sessionId=1591563903)
```

Como en este proyecto todas las entidades van a compartir el mismo fichero de logs, se podrá observar fácilmente el protocolo mensaje a mensaje, conociendo qué entidades los mandan y cuáles los reciben.

Cabe destacar que el registro se divide según la red social en la que se dé, es decir, cuando el usuario inicie sesión en *Friendbook* (lo cual se indica mediante parámetro con el identificador "1"), los mensajes se van a almacenar en el directorio "/logs/Friendbook", mientras que en caso contrario se hará en "/logs/MyLeisure".

Este módulo tiene una segunda función, la de registrar los errores sucedidos durante la ejecución de la aplicación.

Ésta función se va a llevar a cabo mediante el método "error()", que se va a encargar tanto de generar los ficheros de registro de errores como de generar y almacenar los mensajes que detallen la incidencia.

El funcionamiento de este método es igual al de "log()". En primer lugar genera el fichero de registro de errores, dándole como nombre la fecha (día-mes-año) del sistema. En este fichero, va a incluir un mensaje formado por la fecha del sistema, indicando la hora exacta de éste, y el mensaje recibido por la entidad correspondiente en el que se indicará el error surgido.

Gracias a este segundo registro se facilita la tarea de mantenimiento de la aplicación.

3.2 Diagramas de secuencia

Una vez se han detallado las entidades participantes y los servlets que se ocupan de su funcionamiento en el momento de interconectar, se procede a la exposición de los diagramas de secuencia. Estos diagramas muestran de una manera visual las acciones que se llevan a cabo para interconectar cada una de las entidades y, por tanto, cada uno de los servlets definidos.

Antes de comenzar la presentación, es importante destacar que en las llamadas a los métodos que aparecen en los diagramas tan solo se mencionan los parámetros que incluyen en el caso de que existan campos variables, omitiendo los campos fijos.

La exposición de estos diagramas se divide en tantas partes como acciones se puedan llevar a cabo dentro de la red social que requiera algún tipo de interconectividad

Inicio de sesión:

En el siguiente diagrama se puede observar el proceso de inicio de sesión de un usuario.

En el protocolo, este proceso se encuentra unido a la obtención del perfil del usuario que ha iniciado la sesión, pero con el fin de facilitar la comprensión de los diagramas, se ha divido en dos partes. Por este motivo, el flujo no finaliza en un JSP, ya que es necesaria esta segunda parte para poder mostrar datos.

Cabe destacar que este proceso se realiza de manera interna en los servidores de las redes sociales, por lo que el objetivo de este diagrama es comprender las conexiones realizadas entre los distintos elementos de estas redes.

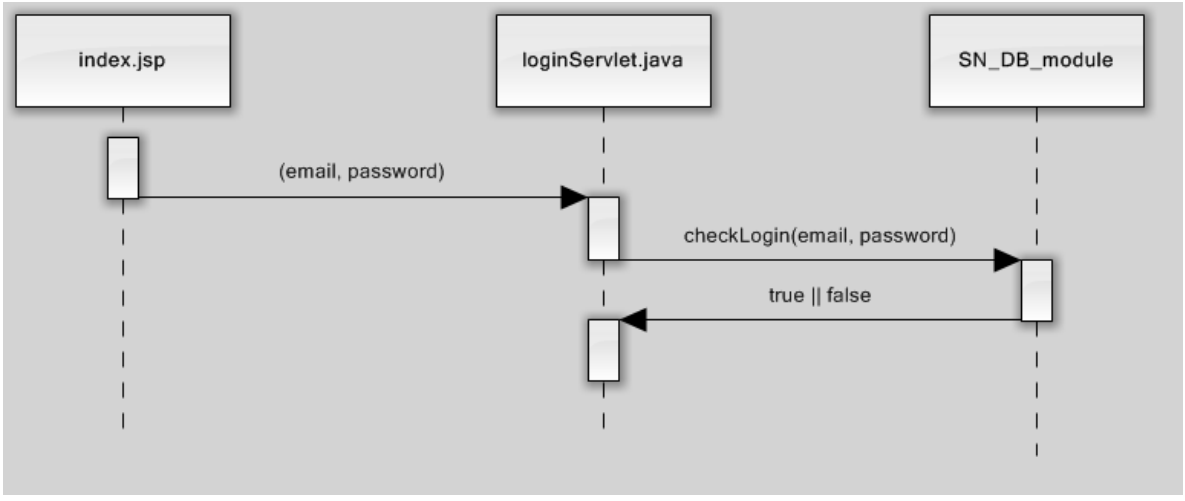

Ilustración 51: Inicio de sesión

Index.jsp es el encargado de proporcionar al usuario el cuadro de login. Una vez el usuario introduce los campos requeridos (email y password), y pulsa el botón correspondiente para iniciar la sesión, el JSP manda estos datos al servlet encargado de verificar que son correctos.

loginServlet.java, una vez recibe dichos datos, verificará que son correctos. Para ello, obtiene el hash de ambos campos y realiza una llamada al módulo encargado de conectar con la base de datos de la red social. Este módulo, mediante el método "checkLogin()", comprobará todos los registros de la base de datos hasta encontrar el hash email del usuario. Una vez localizado, comprobará que el hash de la contraseña almacenado junto a él se corresponde con el proporcionado por el usuario. El resultado obtenido lo devuelve al servlet, paso en el cual se finaliza la primera parte del protocolo.

Obtención del perfil del usuario:

Este segundo diagrama corresponde con la continuación del inicio de sesión. En este caso, actuarán diversas entidades dado que se debe realizar todo el proceso de autenticación del usuario para poder acceder a la información del perfil.

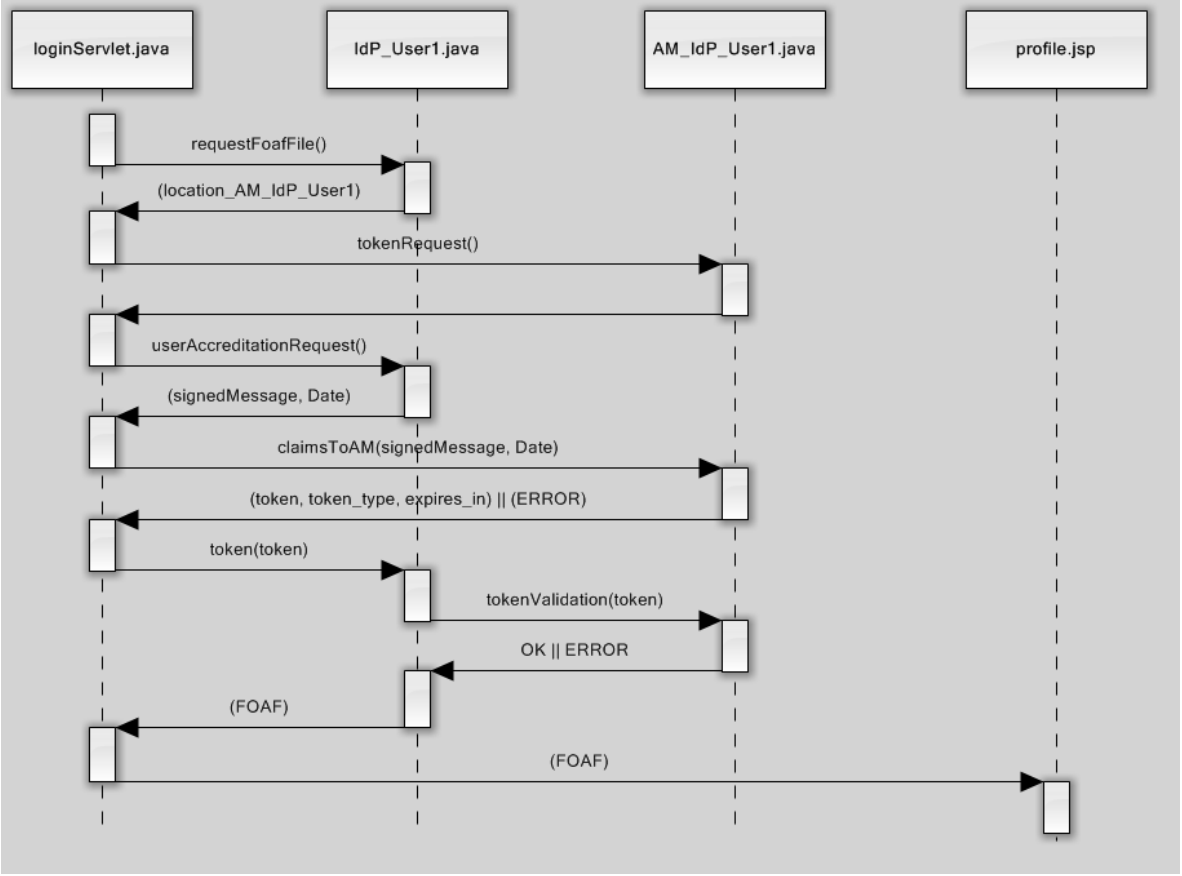

Ilustración 52: Obtención del perfil del usuario

El flujo comienza en el punto en el que se quedó en el diagrama anterior, *loginServlet.java*. Si el resultado del primer apartado ha sido correcto, comenzará el protocolo para obtener los datos del perfil del usuario, enviando los mensajes descritos en la Sección 2.1.1.3 gracias a los métodos descritos en la Sección 3.1.

Una vez finalizado el protocolo con éxito, el servlet se encargará de proporcionar a *profile.jsp* el fichero FOAF que contiene el perfil del usuario. Este último JSP extraerá la información de este fichero, mostrando en su interfaz el perfil completo del usuario.

Obtención de recursos del usuario:

Como se ha comentado en diversas ocasiones, el proceso de obtención de recursos del usuario que ha iniciado la sesión es muy parecido al de obtención de su perfil. Aun así, es necesario mostrar un diagrama con este proceso ya que existen algunas diferencias, principalmente, el cambio de entidades participantes (*IdP* es sustituido por *Host*, y se cambian del mismo modo los *AM*).

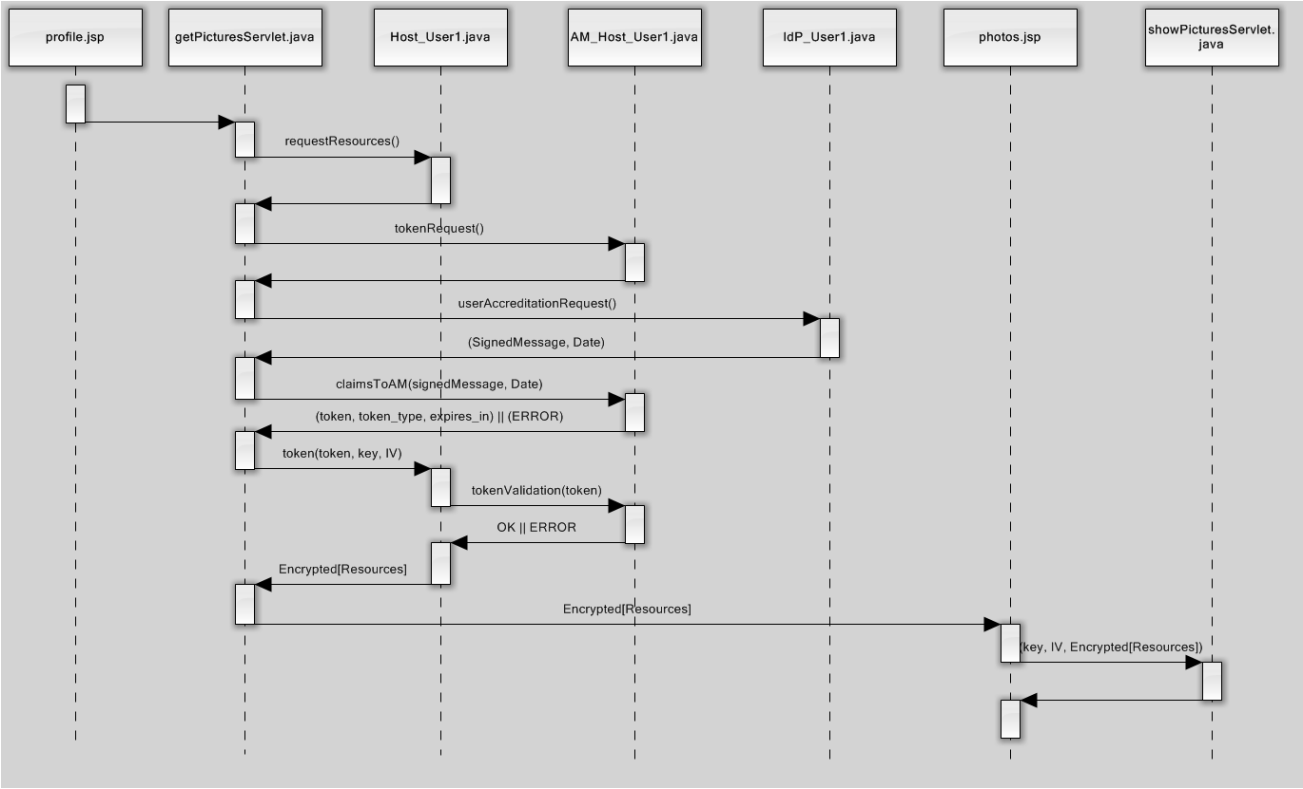

Ilustración 53: Obtención de recursos del usuario

Como se puede observar en el diagrama, el flujo de esta parte del protocolo comienza en *profile.jsp*, concretamente en el momento en que el usuario pulsa el botón "Fotos". En este momento se llama al servlet *getPicturesServlet.java*, que como se ha detallado en la Sección 3.1, es el encargado de realizar el proceso de obtención de recursos.

Si todo el protocolo se cumpliese con éxito, este mismo servlet ha de recibir los recursos (fotografías) del usuario, junto a la clave y el vector de inicialización necesarios para descifrarlos. Con el fin de mostrarlas al usuario a través de la interfaz, se mandan al JSP *photos.jsp*, que se apoyará en *showPicturesServlet.java* para descifrarlas y mostrarlas correctamente.

Visualización del listado de contactos (MyLeisure):

De nuevo se describe un proceso interno de las redes sociales y, éste en concreto, aparece únicamente en *MyLeisure* dado que en *Friendbook* se omite al mostrar la lista de contactos junto a los datos del perfil.

El proceso es simple y tan solo participan los elementos que componen esta entidad.

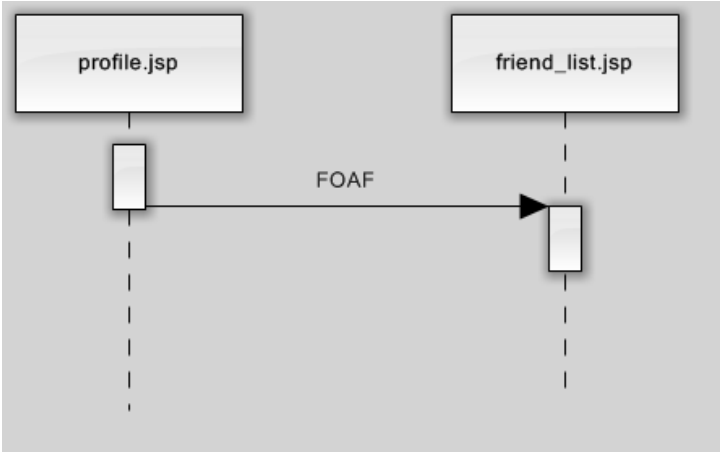

Ilustración 54: Visualización del listado de contactos (MyLeisure)

En el momento en el que el botón "Amigos" del perfil del usuario es pulsado, se realiza una redirección hacia *friend_list.jsp*, el cual recupera la lista de contactos del fichero FOAF (almacenado en la sesión) y la muestra a través de la interfaz.

Visualización del listado de redes sociales:

Este proceso se lleva a cabo en ambas redes sociales y, como se puede observar en el siguiente esquema, se realiza de manera interna en el servidor de cada una de las redes sociales.

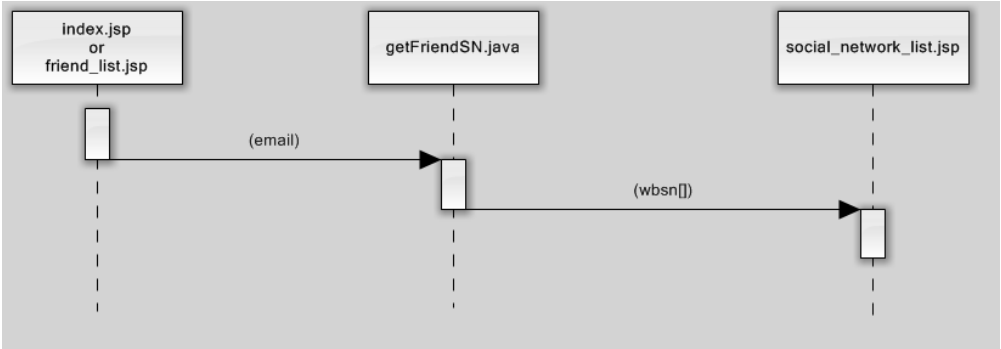

Ilustración 55: Visualización del listado de redes sociales

Al pulsar sobre el nombre de alguno de los contactos, el flujo se dirige hacia *getFriendSNServlet.java*, que se encarga de extraer del fichero FOAF la lista de redes sociales en las que el contacto se encuentra registrado y se las proporciona a *social_network_list.jsp*. Este JSP reconocerá el nombre de cada elemento de la lista, y mostrará el logo correspondiente a la red social leída.

Obtención del perfil de un contacto:

Proceso correspondiente a la parte del protocolo definida en la Ilustración 56. En esta Ilustración se observaba el intercambio de mensajes entre las diferentes entidades, mientras que en diagrama que se observa a continuación se profundiza más, observando los servlets, JSP y métodos que interactúan.

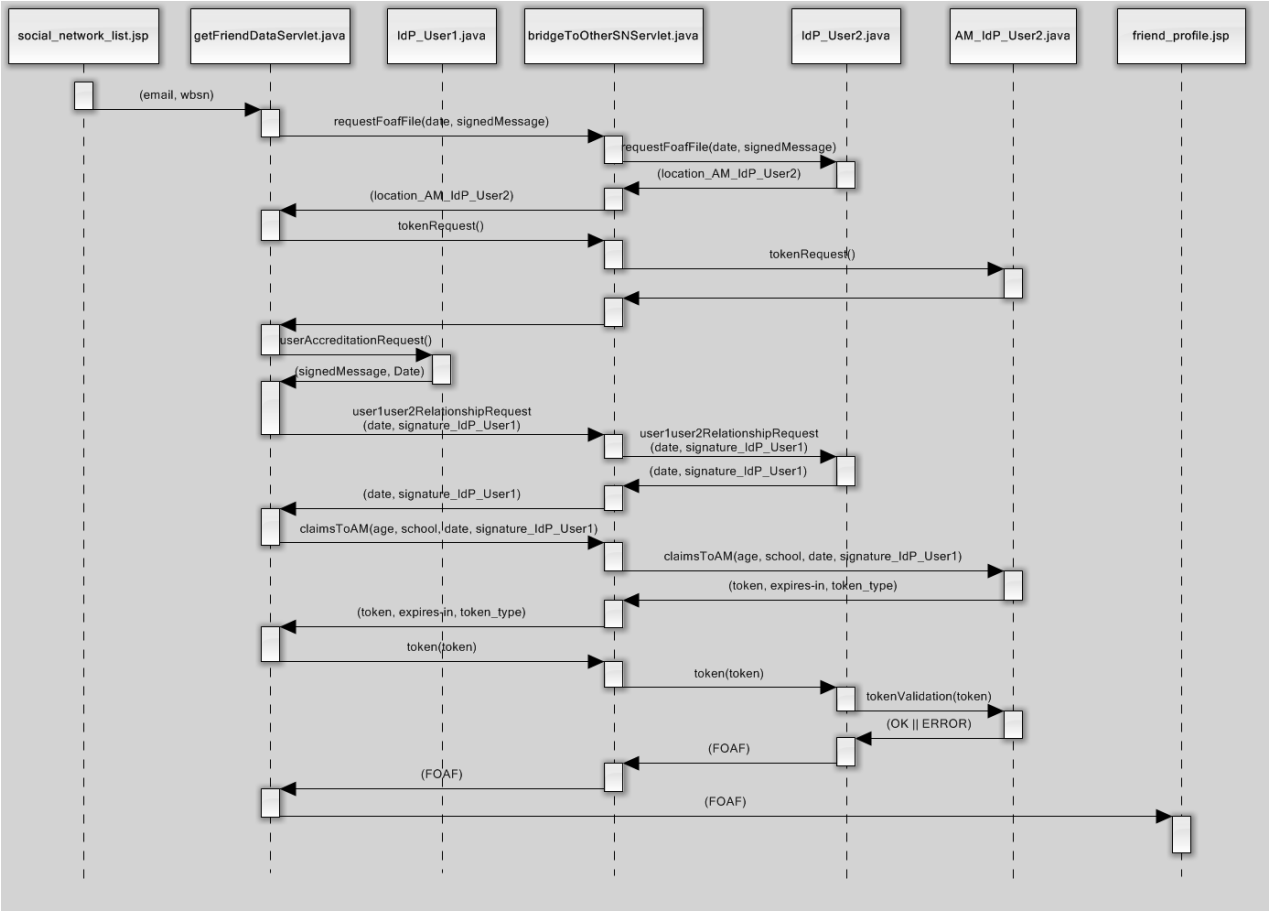

Ilustración 56: Obtención del perfil de un contacto

Como se puede observar, el funcionamiento es similar al de obtener los datos del perfil del usuario, con la diferencia de los mensajes intercambiados, los cuales se detallan en la descripción del protocolo.

En este caso el flujo comienza en el JSP encargado de mostrar la lista de redes sociales de un contacto, que manda la red social seleccionada al servlet encargado del proceso.

Cuando el protocolo finaliza, se proporciona el FOAF del contacto a otro JSP, *friend_profile.jsp*, que mostrará el perfil del contacto.

Obtención de los recursos de un contacto:

En el siguiente diagrama se detalla el proceso de obtención de recursos de un contacto, similar al de la obtención de datos del perfil de éste, pero con ciertas diferencias, como se puede comprobar a continuación.

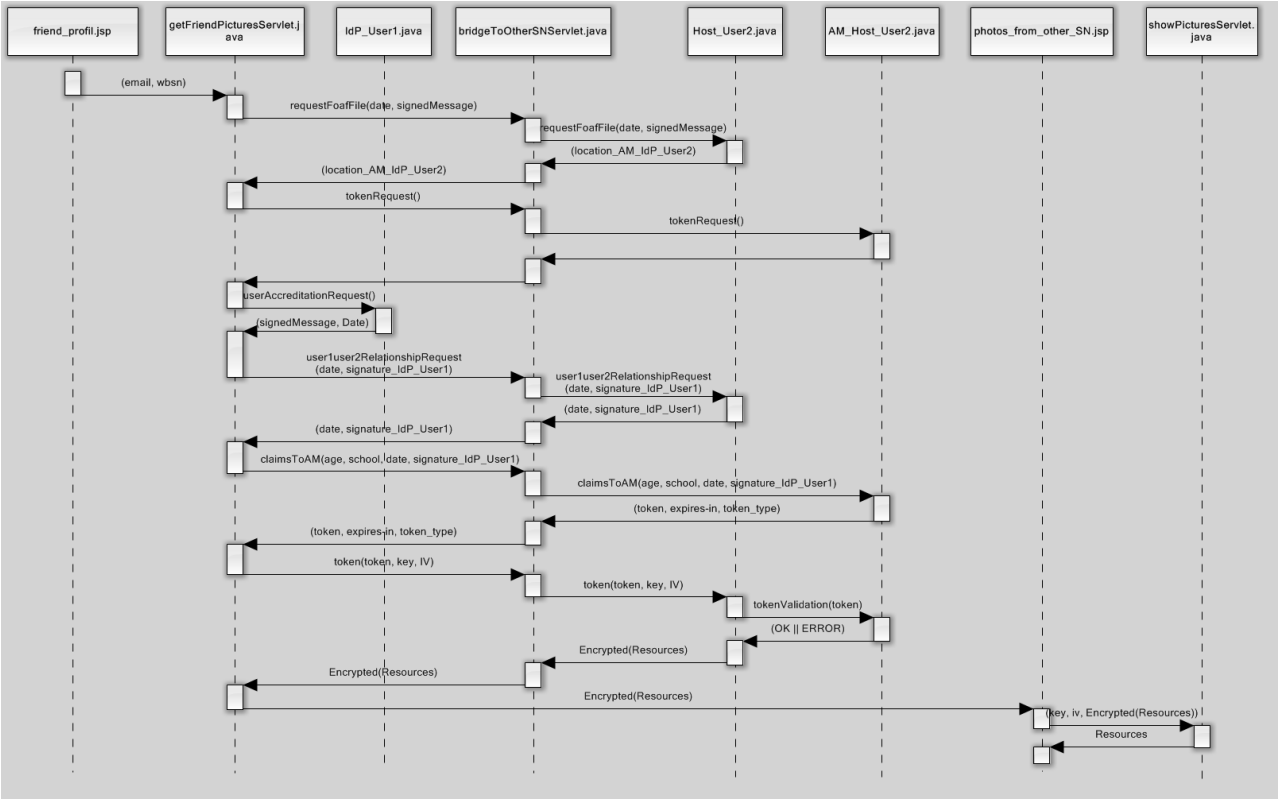

Ilustración 57: Obtención de los recursos de un contacto

Diagrama similar al comentado anteriormente, con la diferencia de que al final del protocolo se obtendrán las fotografías del usuario en lugar del perfil. Para finalizar, un JSP, *photos_from_other_SN.jsp*, se apoya en el servlet *showPicturesServlet.java* para descifrar las imágenes y mostrarlas correctamente.

Capítulo 4

– Implementación y pruebas –

4.1 Implementación

En esta sección van a comentar los detalles de la implementación que por algún motivo cobran cierta importancia.

4.1.1 Enlaces a la aplicación

En primer lugar hay que mencionar la forma a través de la cual se accede a la aplicación una vez finalizada su implementación. Este acceso se realiza a través de un navegador web, accediendo a una de las dos siguientes URLs, en función de cuál de las dos redes sociales se desee visitar:

- http://89.141.92.66:48605/FriendBook/
- http://89.141.92.66:26811/MyLeisure/

En el caso de que se accediese a *Friendbook*, los datos con los que se debe realizar el login son alicia@gmail.com como email y "user1pass" como password. Si la red social visitada fuese *MyLeisure*, se debe usar Benito@gmail.com como email y "user2pass" como password.

Cabe destacar que ambas web se ejecutan sobre un servidor personal, por lo que es probable que en algunos momentos se encuentre caído.

4.1.2 Navegabilidad entre interfaces

Conociendo cada una de las interfaces del proyecto descritas en la Sección 2.1, así como los servlets que actúan entre cada una de ellas, se va a concretar la navegabilidad entre estos componentes al realizar las acciones que permiten cada una de las interfaces.

Para ello, dada la similitud en la parte lógica de ambas redes sociales, se va a tomar una de ellas como ejemplo, Friendbook. No obstante, en las siguientes imágenes se presentan las páginas de inicio de cada una de las redes sociales.

Ilustración 58: Página principal de Friendbook

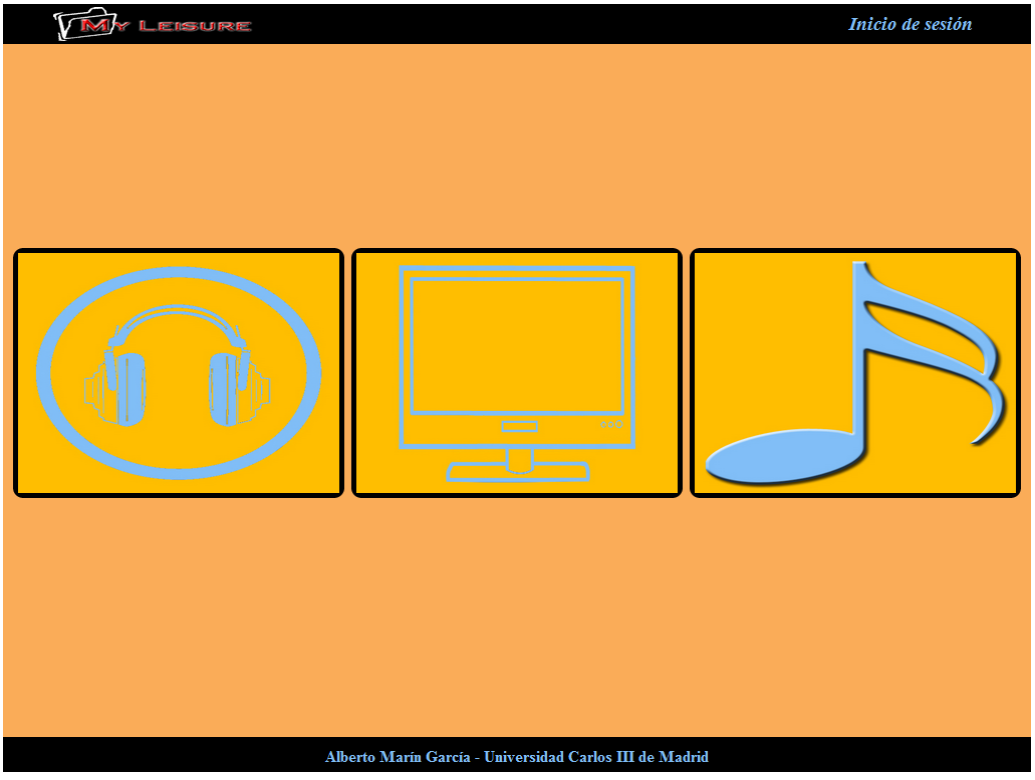

Ilustración 59: Página principal de MyLeisure

Es importante mencionar en la navegabilidad que se describen a continuación se considera que los procesos intermedios han sido satisfactorios. En caso contrario, la aplicación retorna a la página en la que se encontraba previamente.

Al comenzar a usar la red social, se presenta la página principal (ventana izquierda de la Ilustración 60), en la que aparece el cuadro de login para que el usuario pueda iniciar su sesión. En esta página se deben insertar los datos correctos, y pulsar el botón indicado para iniciar la sesión. Al realizar esta acción, se ejecuta el servlet "loginServlet.java", que tras comprobar si los datos son correctos, obtiene el perfil del usuario y dirige el flujo hacia la página de perfil (ventana derecha de la Ilustración 60).

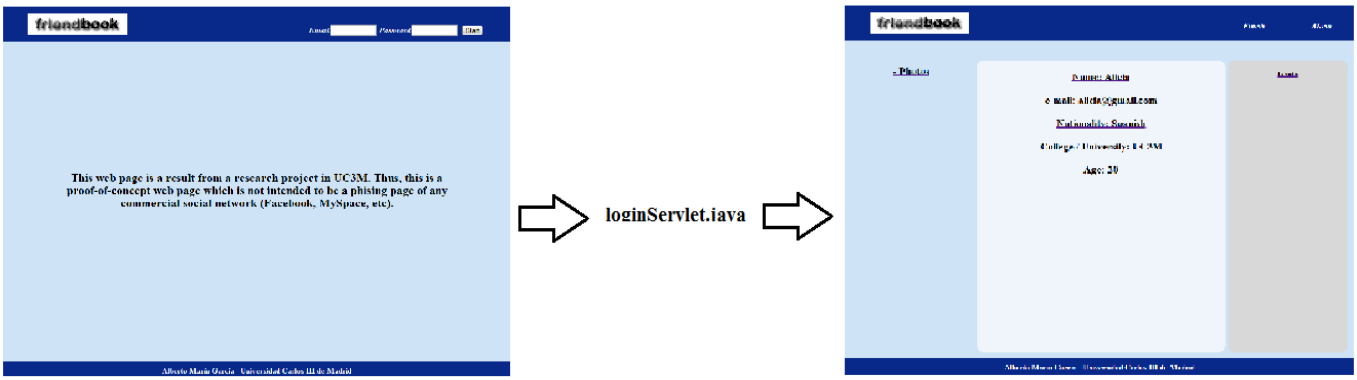

Ilustración 60: Navegabilidad – Inicio de sesión

Desde la página del perfil, se permite realizar dos acciones al usuario. Por un lado, acceder a las fotografías que mantiene almacenadas en el servidor correspondiente, como se muestra en la Ilustración 61, y por otro lado visualizar el perfil de alguno de sus contactos, escenificado en la Ilustración 62. En el primer caso, al pulsar sobre el texto "Photos", pasa a ejecutar "getPicturesServlet.java", para obtener las fotografías del usuario. Una vez tiene las imágenes en su poder, dirige el flujo hacia "photos.jsp", que se apoyará en el servlet "showPicturesServlet.java", para mostrar todas las imágenes del usuario.

Ilustración 61: Navegabilidad – Obtención de recursos del usuario

Si la opción tomada por el usuario fuese la de ver el perfil de alguno de sus contactos, debería pulsar sobre el nombre de alguno de estos. Tras esto, se ejecutaría "getFriendSN.java", que recupera el listado de redes sociales en las que mantiene un perfil este contacto y llama a "social_network_list.jsp" para que lo muestre por pantalla.

Al seleccionar una de las redes sociales del listado, pasa a ejecutar "getFriendDataServlet.java", que se encarga de recuperar el perfil del contacto seleccionado y dirigir el flujo hacia la página correspondiente para mostrar el perfil de un contacto.

Ilustración 62: Navegabilidad – Obtención del perfil de un contacto

Desde el perfil del contacto, se ofrece la posibilidad de acceder a las fotografías de éste. Para ello, de nuevo se debe pulsar sobre "Photos", y en este caso el servlet que entra en ejecución es "getFriendPicturesServlet.java". El proceso que se realiza es similar al de obtener los recursos del propio usuario, dado que este servlet recuperará las fotografías del contacto, dirigirá el flujo a la página encargada de mostrar las fotos, y ésta se apoyará en "showPicturesServlet.java" para mostrarlas correctamente. En la siguiente imagen se muestra el resultado.

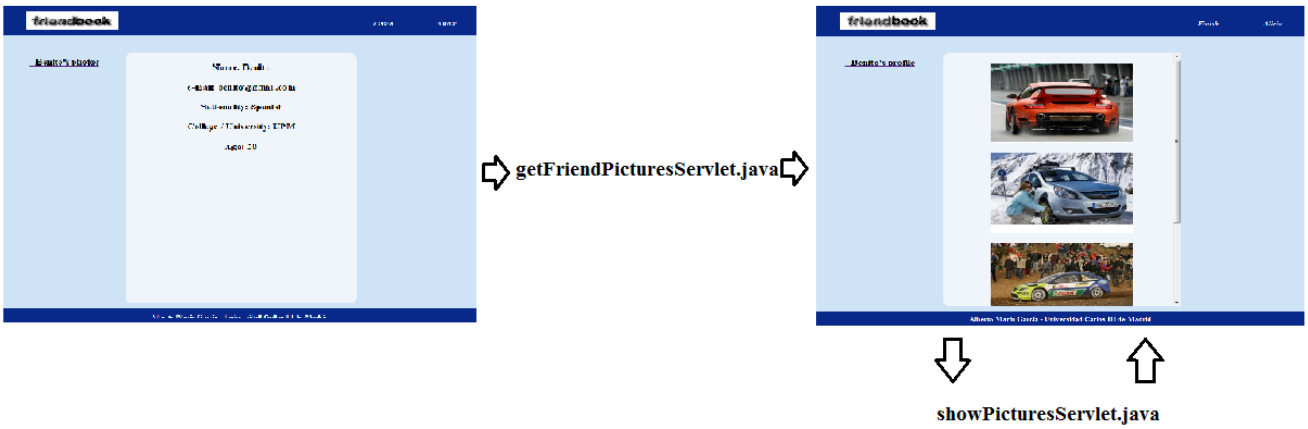

Ilustración 63: Navegabilidad – Obtención de recursos de un usuario

Por ultimo mencionar que todas las páginas, una vez se ha iniciado la sesión, ofrecen la posibilidad de finalizarla. Para ello, se debe pulsar el botón "Finish session", momento en el cual se ejecutará "finishSessionServlet.java", que eliminará los datos de la sesión actual y redirigirá la aplicación a la página principal.

4.1.3 Gestión, transferencia y almacenamiento de claves

El hecho de que para realizar el cifrado y descifrado de las claves simétricas con las que se cifraron previamente las imágenes deba existir intercambio de claves entre el usuario y el host, introduce un problema de implementación en varios ámbitos.

Como ya se ha comentado, las imágenes se almacenarán cifradas a través de un algoritmo de clave simétrica y, para poder descifrarlas posteriormente, se debe almacenar también dicha clave simétrica. En este momento surge la primera dificultad, y es la elección del formato al que habrá que convertir la clave simétrica para poder almacenarla en la base de datos. Al generar la clave, se hace a través de un objeto "SecretKeySpec", específico de JAVA y perteneciente a la librería "security" de este lenguaje. "SecretKeySpec" posee un método gracias al cual se puede transformar este objeto en un array de bytes, tipo de datos mucho más común y manejable. Por tanto, se realizará una llamada a este método (getEncoded()), y se almacenará la clave en dicho array, que generalmente recibirá el nombre de "simetricKeyBytes", para diferenciarlo del resto de tipo de objetos. Obtenido el Array, se puede proceder a almacenar la clave simétrica en la base de datos, en un campo de tipo BLOB.

Por otro lado, para realizar el cifrado de esta llave simétrica, desde la red social se debe proporcionar al host una clave pública con la que realizar el cifrado. En este momento surge la segundo complicación, basada en el cómo transferir esta clave desde un servidor a otro. Gracias a la solución comentada anteriormente, se puede pasar la clave pública a un Array de bytes. Para finalizar, se transforma dicho array en un String, formato idóneo para transferir entre servidores.

4.1.4 Algoritmos de cifrado. Selección del tamaño de claves

Conociendo como se va a realizar la transferencia y el almacenamiento de las claves para el cifrado, se procede a especificar cuáles son los algoritmos de cifrado y, más concretamente, que tamaño de clave se va a usar en cada uno de ellos.

En primer lugar, para el cifrado de clave simétrica, se opta por el algoritmo AES. Para este cifrado se usará una clave de 128 bits. Gracias a la característica que conlleva la clave simétrica, se hará uso de una única clave para el cifrado y el descifrado. De esta forma, una vez se realiza el cifrado de la imagen para almacenarla en la base de datos, se puede almacenar esta clave junto a ella para poder descifrarla posteriormente.

Por otro lado, se opta por el algoritmo RSA para realizar el cifrado asimétrico. Este algoritmo puede utilizar distintos tipos de claves, como las propias de RSA o Diffie-Hellman. En este caso se genera un par de claves RSA, con un tamaño de 2048 bits. A partir de este par de claves (objeto de la clase "KeyPair"), se podrá obtener tanto la clave pública como la privada.

Junto al tamaño de clave y el algoritmo seleccionado, hay que mencionar que el modo de cifrado seleccionado es el "ECB", y se usará un padding de tipo "PKCS1Padding".

Con la selección de estos algoritmos y sus correspondientes tamaños de claves, se calcula que la seguridad de la aplicación tendrá una duración relativamente larga, dado que actualmente no se conoce la posibilidad de romperlos en un tiempo aceptable.

De cualquier modo, como se ha comentado durante el documento, tanto los algoritmos como los tamaños de claves se podrán modificar posteriormente, por lo que la aplicación se puede adaptar a las diferentes vulnerabilidades que vayan surgiendo en dichos algoritmos.

Capítulo 5

–Evaluación –

5.1 Verificación de pruebas

Tras la finalización del desarrollo del sistema, se procede a realizar las pruebas establecidas en el Plan de Pruebas que se puede encontrar en la Sección 2.7.

El cumplimiento de todas estas pruebas supone que la aplicación cumple con éxito todas las funcionalidades establecidas en los requisitos del sistema, con un funcionamiento correcto y sin que se produzcan fallos durante la ejecución.

En la siguiente tabla, se observa el estado de cada una de las pruebas:

Verificación de pruebas		
Identificador	**Versión**	**Resultado**
PA-01	1.0	ACEPTADA
PA-02	1.0	ACEPTADA
PA-03	1.0	ACEPTADA
PA-04	1.0	ACEPTADA
PA-05	1.0	ACEPTADA
PA-06	1.0	ACEPTADA
PA-07	1.0	ACEPTADA
PA-08	1.0	ACEPTADA
PA-09	1.0	ACEPTADA

Tabla 20: Verificación de pruebas

5.2 Evaluación de los resultados

A continuación se va a realizar una evaluación del funcionamiento de la aplicación, presentando los tiempos obtenidos en diferentes mediciones. Estas mediciones se han realizado en varios puntos concretos: inicio de sesión del usuario y obtención de su perfil, obtención de los recursos del usuario que inicia la sesión, obtención del perfil de uno de los contactos y obtención de los recursos de dicho contacto.

Para obtener unos tiempos lo más similares posibles a los que se conseguirían al implantar cada entidad del proyecto en un servidor, se deberían haber realizado las pruebas implantando cada entidad en un ordenador distinto y conectado a una red distinta (dado que existen diez entidades, serían necesarios diez ordenadores en diez redes distintas). Debido a que actualmente el desarrollador no dispone de esta infraestructura, se han realizado las pruebas implantando el proyecto sobre tres equipos, cada uno de ellos sobre una red distinta, repartiendo las entidades de la siguiente manera:

- **Equipo 1:** redes sociales y *AMs*.
- **Equipo 2:** *IdPs*.
- **Equipo 3:** *Hosts*.

Es posible pensar que dada la infraestructura en la que se realizan las pruebas, debido a que se envían sus mensajes en local, los resultados obtenidos van a ser mejores que en un entorno real. Sin embargo, se ha de indicar que en entorno real se utilizarían servidores dedicados que, al contrario que en esta evaluación, producirían un mayor rendimiento.

Conociendo la situación en la que se realizan las medidas, se procede a exponer los tiempos tomados para cada una de las acciones comentadas. Estos tiempos se tomarán hasta en cinco ocasiones para cada una de las acciones. De esta forma, se puede calcular un valor medio que proporcione una mayor fiabilidad que realizar tan solo una medida.

Con el fin de tener un referente a la hora de evaluar los tiempos obtenidos, se van a realizar medidas en otra red social real, en este caso Facebook, y se mostrarán junto a los tiempos de este proyecto.

Inicio de sesión y recuperación del perfil de un usuario:

Prueba	Tiempo del proyecto	Tiempo de Facebook
1	2’213 s	2’45 s
2	2’605 s	2’12 s
3	3’298 s	1’98 s
4	2’903 s	3’61 s
5	2’437 s	2’60 s
MEDIA	**2’691 s**	**2’552 s**

Tabla 21: Evaluación – Inicio de sesión y recuperación del perfil

El tiempo medio consumido en el inicio de sesión es de 2’691 segundos, lo cual supone una espera corta que no afecta al usuario, y un tiempo equiparable al que se puede obtener al iniciar sesión en otra red social como Facebook, en la que los datos se mantienen en un mismo servidor.

Obtención de recursos del usuario:

Prueba	Tiempo del proyecto	Tiempo de Facebook
1	3’891 s	2’15 s
2	4’251 s	2’30 s
3	3’817 s	3’21 s
4	5’544 s	2’86 s
5	3’951 s	2’62 s
MEDIA	**4’29 s**	**2’628 s**

Tabla 22: Evaluación – Obtención de recursos del usuario

Pese a que el número de mensajes intercambiados, tal y como define el protocolo, es el mismo que en la acción anterior, se obtienen tiempos mayores ya que se realiza el cifrado y descifrado de una clave simétrica, y el descifrado posterior de las fotografías, lo que supone un proceso más pesado. Aun así, el tiempo obtenido es aceptable, dado que en cualquier sistema el proceso de carga de imágenes es más lento que el de textos.

En este caso, la aplicación desarrollada es algo más lenta que Facebook, si bien es cierto que mientras en la simulación de red social se cargan directamente las imágnes con su tamaño real, Facebook carga miniaturas de éstas, lo que las hace menos pesadas.

A pesar de todo, esta diferencia de tiempo no es lo suficientemente grande como para conseguir que un usuario deciese no hacer uso de la aplicación.

Obtención del perfil de un contacto:

Prueba	Tiempo del proyecto	Tiempo de Facebook
1	2’604 s	2’31 s
2	2’494 s	2’71 s
3	2’361 s	2’54 s
4	2’582 s	3’69 s
5	2’92 s	2’85 s
MEDIA	**2’592 s**	**2’82 s**

Tabla 23: Evaluación – Obtención del perfil de un contacto

En esta ocasión se obtiene el tiempo más rápido de los calculados, aunque muy similar al obtenido en el proceso de inicio de sesión. El hecho de obtener un mismo tiempo se debe a que pese a que en esta parte del protocolo se envían una mayor cantidad de mensajes, tan solo se realiza una consulta en base de datos (obtener el perfil) Sin embargo, en el proceso de login, con una menor cantidad de mensajes enviados, se realizan dos consultas a la base de datos (chequear usuario y password y obtener el perfil).

La comparación con Facebook muestra tiempos muy similares, lo que demuestra que la aplicación desarrollada es eficiente en los procesos de carga.

Obtención de recursos de un contacto:

Prueba	Tiempo del proyecto
1	10’664 s
2	10’344 s
3	11’16 s
4	11’379 s
5	11 s
MEDIA	**10’909 s**

Tabla 24: Evaluación - Obtención de recursos de un contacto

Como se puede comprobar, este proceso es el más largo de los cuatro llevados a cabo. Ésta amplía diferencia de tiempos se debe a que esta parte del protocolo es la que más cantidad de mensajes contiene. Además, en varios de estos mensajes se envían fotografías, lo que los hace más pesados y el tiempo de envío aumenta. Por otro lado, se realiza cifrado y descifrado de la clave simétrica y, posteriormente, el descifrado de cada una de las imágenes. A pesar de esto, el tiempo obtenido se considera aceptable y cumple con las restricciones establecidas en los requisitos, donde se determina que el tiempo máximo para completar esta acción es de 15 segundos (RNF-04).

En esta evaluación no se han tomado tiempos en Facebook ya que no ofrece la posibilidad de recuperar todas las imágenes de un contacto desde el perfil, sino que muestra en primer lugar un listado de álbums.

Para finalizar la sección, se presentan los gráficos de los tiempos obtenidos en cada una de las cuatro acciones, con el fin de poder compararlos de una manera más visual.

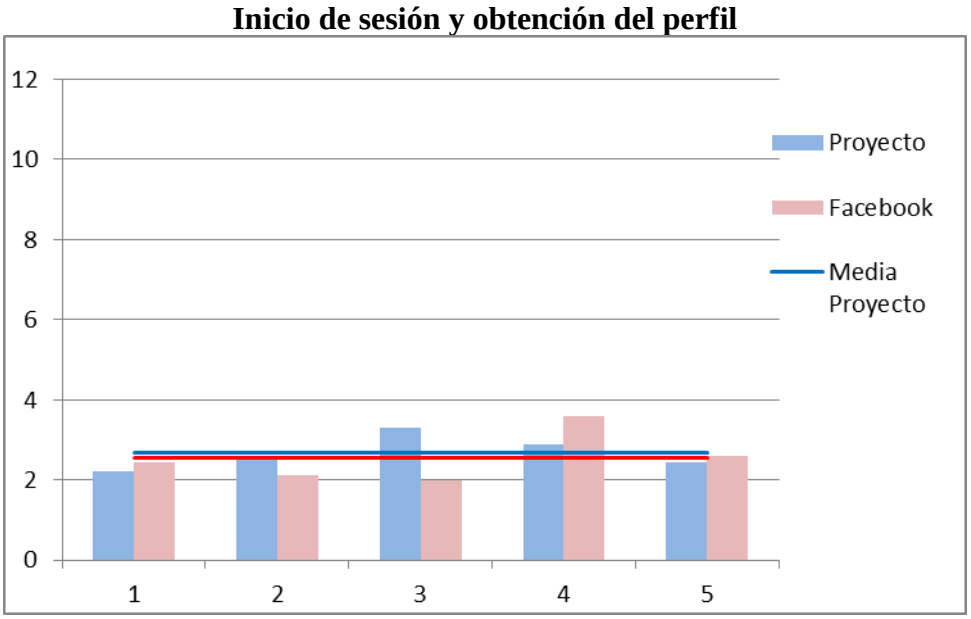

Ilustración 64: Gráfica de tiempos – Inicio de sesión y obtención del perfil

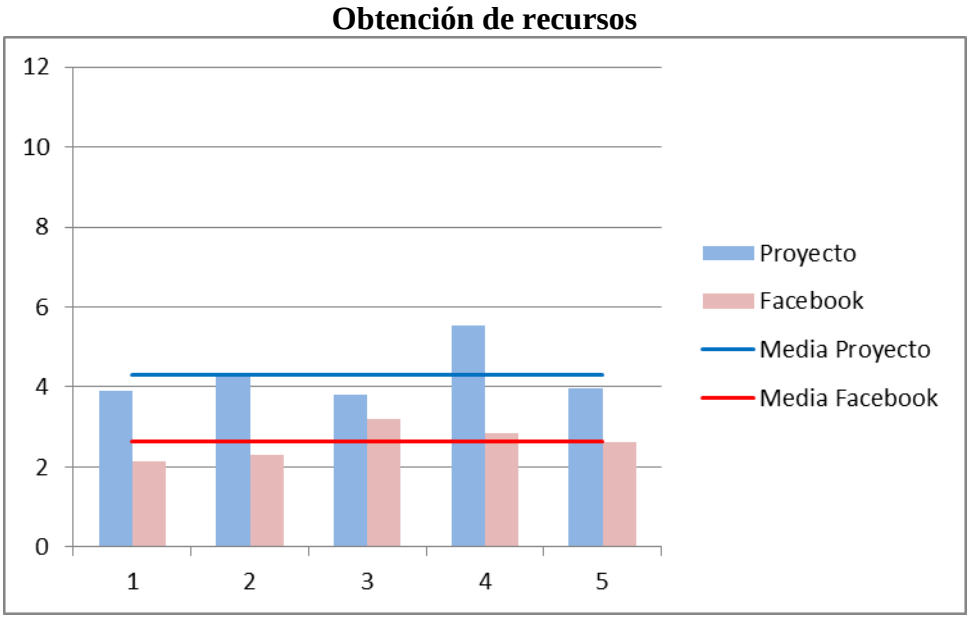

Ilustración 65: Gráfica de tiempos – Obtención de recursos

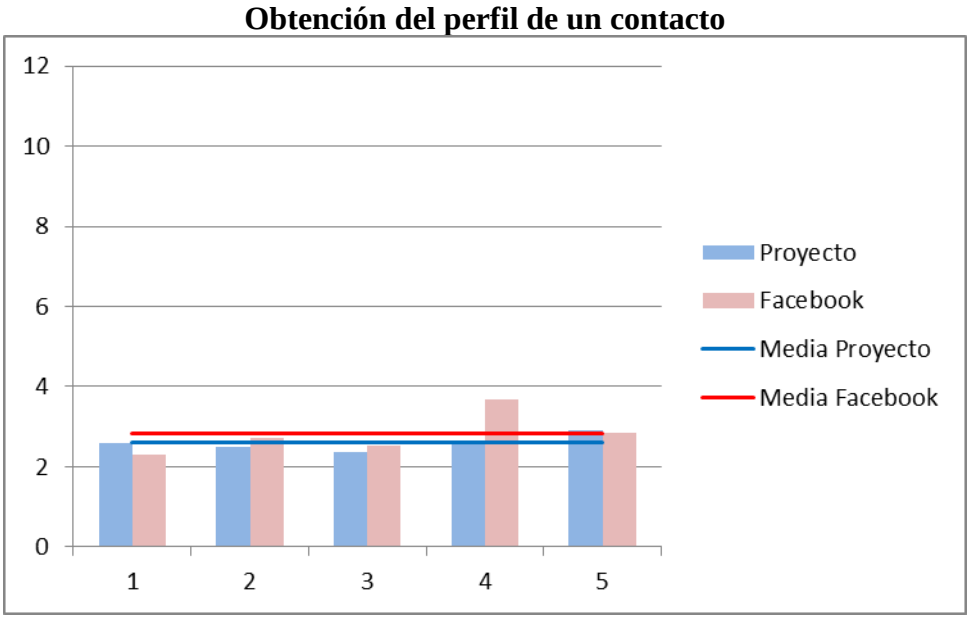

Ilustración 66: Gráfica de tiempos – Obtención del perfil de un contacto

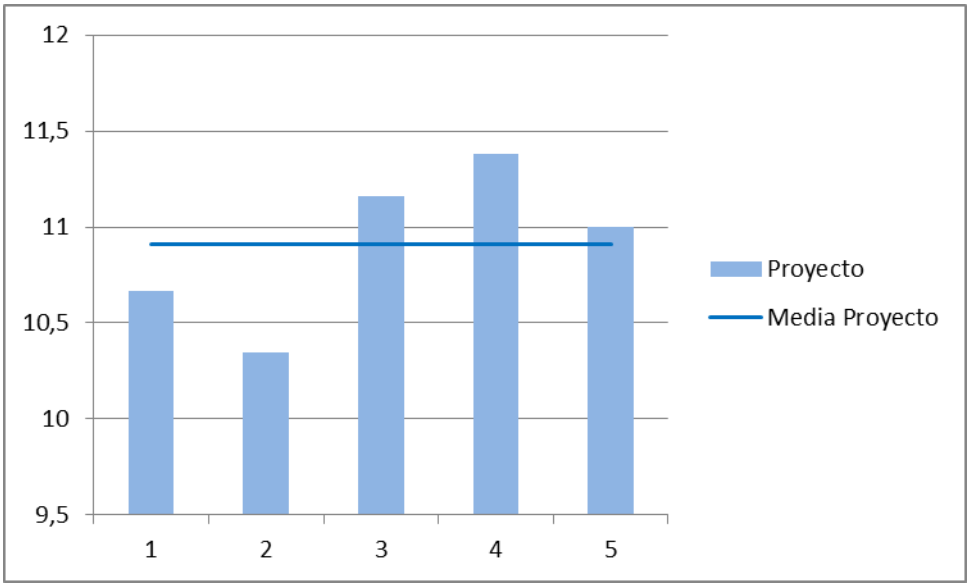

Ilustración 67: Gráfica de tiempos – Obtención de recursos de un contacto

Capítulo 6

– Conclusiones y líneas futuras –

6.1 Conclusiones

La principal conclusión que se extrae de la realización de este proyecto es la importancia de los datos privados de los usuarios de una aplicación, de mantenerlos seguros e inaccesibles a agentes externos, ya sean atacantes como las propias redes sociales, tanto en su almacenaje en bases de datos como en los intercambios de mensajes entre distintos servidores.

Este hecho conlleva que el proyecto desarrollado tenga una gran importancia y sea innovador en relación con el uso de las redes sociales. Por un lado, las redes sociales manejan los datos de forma centralizada, impidiendo la interoperabilidad de los mismos. Por otro lado, aunque las redes sociales protegen la información de los usuarios, en mayor o menor medida, no existe un agente externo que la proteja de estas mismas redes sociales. Por estos motivos, con este proyecto se consigue proteger la información de los usuarios centralizándola para conseguir interoperabilidad y aplicando técnicas criptográficas para impedir que grandes empresas, propietarias de las distintas redes sociales, hagan un uso indebido de la información que disponen.

Por otro lado, el hecho de tratarse de una aplicación distribuida permite conocer cómo comunicar diferentes servidores de manera eficiente, pese a que tengan que actuar multitud de estos servidores. Como se observa en la Sección 5, se han conseguido unos tiempos de ejecución equiparables a los que se encuentran en las actuales redes sociales, lo que conlleva que el proyecto sea aplicable al uso general en las redes sociales, sin perjudicar la usabilidad de los clientes de estos servicios.

Para concluir, el hecho de afrontar un proyecto completo de manera individual, desde su planificación hasta la realización de las pruebas finales, aporta grandes conocimientos.

En la elaboración de la planificación se aprende a realizar estimaciones del tiempo requerido para cada tarea, cualidad de gran importancia para futuras ocasiones. En la fase de análisis, el hecho de considerar cual debe ser la arquitectura que se debe seguir y las tecnologías que hay que usar para la realización del proyecto, evaluando los pros y los contras de cada una de ellas, aporta nuevas capacidades en cuanto a la toma de decisiones. Por último el diseño de la aplicación, donde se especifica cada componente de un sistema aún inexistente, supone una novedad y un nuevo reto que afrontar.

Por tanto, se puede afirmar que el aprendizaje durante el desarrollo de este proyecto ha sido tanto amplio como valioso para el futuro mundo laboral, en que, tomando como punto de partida los conocimientos adquiridos, se deban tomar decisiones en proyectos de mayor envergadura.

6.2 Líneas futuras

Como se ha comentado en diversas ocasiones, existen posibles modificaciones que se podrían aplicar sobre la aplicación aportando diversas mejoras, tanto para el usuario como para el funcionamiento del protocolo.

En primer lugar, con el fin de hacer la aplicación más atractiva para el usuario, se propone la inclusión de otros tipos de recursos para compartir, como pueden ser canciones, vídeos, comentarios de texto, etc. Para que esta mejora sea posible, se deben realizar varios cambios. Por un lado, modificar las bases de datos enlazadas a los *Hosts*, dado que se deben preparar para almacenar el nuevo contenido. Esta modificación podría realizarse creando nuevas tablas y dedicando cada una de ellas a un tipo de contenido o, creando un nuevo campo en la tabla dedicada al almacenamiento del recurso actual, las fotografías. Este nuevo campo debería indicar el tipo de contenido que se almacena, ya sea mediante un texto, un carácter o un número. Dado que se modifica la base de datos, es necesario realizar pequeños cambios en el módulo que la gestiona. Estos cambios dependerán de la opción que se elija a la hora de modificar la base de datos. Es posible que el módulo de cifrado de recursos también fuera afectado por estos cambios, dado que se cambia el tipo de información a tratar. Por último, se deben modificar las interfaces de ambas redes sociales, con el fin de que sean capaces de presentar esta nueva información. La mejor opción que se podría tomar en este caso es la de crear una página para cada tipo de información, de manera que el usuario pueda localizar fácilmente su contenido.

Una segunda mejora que se podría aplicar es la de facilitar a la red social un fichero de configuración en el que se determinase la localización de los servidores externos donde el usuario almacena sus datos. Esta medida mejoraría la seguridad de la información, ya que la red social no conocería la ubicación de la información privada hasta el momento del inicio de la sesión, evitando posible intentos de acceso a ésta. Para aplicar esta medida, se debería en primer lugar modificar las bases de datos de las redes sociales, eliminando de éstas la tabla dedicada a almacenar las localizaciones de los servidores. Por otro lado, la red social debería ser capaz de leer el contenido del fichero y ser capaz de interpretarlo correctamente.

Otra posible medida a tomar es la unificación los dos *AMs* existentes en el protocolo. Llevando a cabo este cambio, se simplificaría la arquitectura del sistema y los ficheros de configuración comentados anteriormente. La aplicación de esta mejora es posiblemente la más sencilla de las comentadas. Tan solo se tendría que permitir al AM interconectar tanto con *IdPs* como con *Host*, y almacenar en su base de datos el contenido que actualmente se encuentra dividido entre *AM_Host_DB* y *AM_IdP_DB*. Cabe destacar que pese a aplicar esta medida, el usuario debe ser el que en última instancia decida si desea utilizar un solo *AM* o dos.

Como última mejora aparece la posibilidad de hacer uso de alguna de las librerías de java llamadas “Base64”. Ésta librería se utilizaría para convertir los datos desde arrays de bytes a String y viceversa. Estas transformaciones son utilizadas en varias ocasiones en la aplicación, principalmente en el uso de imágenes y claves de cifrado, por lo que podría simplificar el código de la aplicación, facilitando su posterior mantenimiento.

Capítulo 7

– Referencias –

En este capítulo se listarán todas las referencias que han aparecido a lo largo del documento. Para elaborar este listado, se va a seguir la Norma ISO 690-2 para documentos electrónicos. Las referencias serán listadas en orden de aparición en el documento.

1. Alexa – The Web Information Company. The top 500 sites on the web. [En línea] http://www.alexa.com/topsites
2. Alexa – The Web Information Company. The top 15 most popular social networking sites. [En línea] http://www.alexa.com/topsites/category/World/Italiano/Computer/Internet/Comunicare/Social_Networking
3. Boletín Oficial del Estado. Ley Oficial de Protección de Datos. [En línea] http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf
4. International Standardization Organization. Information Références bibliographiques. Partie 2: Documents éléctroniques, documents ou parties de documents. Norme international ISO 690-2: 1997 (F). Genève: ISO, 1997, 18p.
5. Últimátum de los consumidores alemanes a Facebook. **Rosalía Sánchez**. [En línea] http://www.elmundo.es/elmundo/2012/08/28/navegante/1346131505.html
6. Facebook. [En línea] https://www.facebook.com
7. Twitter. [En línea] https://twitter.com
8. Badoo. [En línea] http://badoo.com
9. **Joseph Bonneau**, **Sören Preibusch**. The Privacy Jungle: On the Market for Data Protection in Social Networks. [En línea] http://weis09.infosecon.net/files/156/paper156.pdf
10. LinkedIn. [En línea] https://www.linkedin.com
11. Orkut. [En línea] http://www.orkut.com
12. Google+. [En línea] https://plus.google.com
13. Foaf-projects. The friend of a friend. [En línea] http://www.foaf-project.org/
14. MySQL. [En línea] http://www.mysql.com/
15. ESA software engineering standards. Issue 2. [En línea] http://www.fabricadesoftware.cl/documentos/ESA/PSS050.pdf
16. Ministerio de Empleo y Seguridad Social. Régimen General de la Seguridad Social. [En línea] http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm
17. Microsoft Store. Buy Windows 7 Ultimate [En línea] http://www.microsoftstore.com/store/msstore/pd/Windows-7-Ultimate/productID.216647200/parentCategoryID.44066700/categoryID.50726100/list.true&WT.mc_id=WOL_shop
18. NetBeans 7.1 [En línea] http://netbeans.org/
19. GlassFish v3 [En línea] http://glassfish.java.net/public/downloadsindex.html#top
20. Microsoft Store. Buy Office Home and Student 2010. [En línea] http://www.microsoftstore.com/store/msstore/en_US/pd/productID.236555900
21. Microsoft Store. Buy Project Professional 2010. [En línea] http://www.microsoftstore.com/store/msstore/en_US/pd/productID.216568600

22. Software Ideas Modeler [En línea] http://www.softwareideas.net/en/download

Capítulo 8

– Glosario de términos –

En esta sección se va a realizar una descripción de aquellos términos que la necesiten, ya sean siglas, acrónimos, o simplemente términos técnicos de difícil comprensión.

Término	Descripción
AES	**Advanced Encryption Standard**: algoritmo de cifrado por bloques. Tiene un tamaño de bloque fijo de 128 bits y acepta como tamaño de llave 128, 192 o 256 bits.
API	**Application Programming Interface:** conjunto de métodos y procedimientos ofrecidos mediante librerías para ser usados como una capa de abstracción.
Función hash	Función realizada mediante un algoritmo que transforma una cadena de entrada en una de salida de tamaño fijo, que representa a ésta primera y a partir de la cual no se puede recuperar la original
HTTP	**Hypertext Transfer Protocol:** protocolo orientado a transacciones web basado en el esquema petición-respuesta
JDBC	**Java Database Conectivity:** API que ofrece métodos para la interacción con bases de datos en el lenguaje JAVA.
JVM	**Java Virtual Machine:** máquina virtual capaz de interpretar y ejecutar las instrucciones expresadas por el compilador JAVA.
MVC	**Modelo Vista Controlador:** patrón que separa los datos de una aplicación, la interfaz y la lógica de negocio en componentes distintos.
OpenSource	**Código abierto:** software desarrollado y distribuido libremente.
RDF	**Resource Description Framework:** lenguaje para representar información en la web. Desarrollado por el W3C.
RSA	Algoritmo criptográfico de clave pública más utilizado. Permite realizar tanto cifrado como firma.
Servlet	Pequeños programas que ejecutan dentro de un servidor.
SHA-1	Sistema de funciones hash que produce una salida de 20 bytes a partir de un mensaje con un tamaño máximo de 2^64 bits.
Token	Elemento que se entrega a un usuario autorizado para facilitar el acceso en el servicio de autenticación.
URL	**Localizador uniforme de recursos:** secuencia de caracteres de acuerdo a un estándar usado para nombrar recursos en internet.

Tabla 25: Glosario de términos

Apéndice A

– Planificación –

9.1 Planificación inicial

En esta sección se va a presentar la planificación inicial realizada para el proyecto, comparándola con el resumen final de tiempo, para comprobar si se han cumplido los plazos o ha existido desfase en éstos.

Esta planificación previa al inicio del proyecto, estima los días y horas que se necesitarán para realizar cada una de las tareas que lo componen. Para realizar esta estimación, se considera que el trabajo se realizará de lunes a viernes, con un total de seis horas por cada día.

A continuación se observa la tabla de tareas con su correspondiente estimación y el Diagrama de Gantt que la representa de manera gráfica.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		**Proyecto: Control de acceso en redes sociales**	**120 días**	**lun 05/03/12**	**vie 17/08/12**
2		**Planificación**	**3 días**	**lun 05/03/12**	**mié 07/03/12**
3		**Análisis**	**23 días**	**jue 08/03/12**	**lun 09/04/12**
4		Estudio del arte	3 días	jue 08/03/12	lun 12/03/12
5		Estudio tecnológico	3 días	mar 13/03/12	jue 15/03/12
6		Estudio de la arquitectura	4 días	vie 16/03/12	mié 21/03/12
7		Casos de uso	4 días	jue 22/03/12	mar 27/03/12
8		Establecimiento de requisitos	5 días	mié 28/03/12	mar 03/04/12
9		Establecimiento de pruebas	4 días	mié 04/04/12	lun 09/04/12
10		**Diseño**	**29 días**	**mar 10/04/12**	**vie 18/05/12**
11		Diseño de interfaces	5 días	mar 10/04/12	lun 16/04/12
12		Diseño de controladores	9 días	mar 17/04/12	vie 27/04/12
13		Diseño de BBDD y módulo de gestión	5 días	lun 30/04/12	vie 04/05/12
14		Diseño de módulo criptográfico	7 días	lun 07/05/12	mar 15/05/12
15		Diseño de módulo de "logs"	3 días	mié 16/05/12	vie 18/05/12
16		**Implementación**	**62 días**	**lun 21/05/12**	**mar 14/08/12**
17		Implementación de interfaces	7 días	lun 21/05/12	mar 29/05/12
18		Implementación parte 1 del protocolo: Obtención de perfil	8 días	mié 30/05/12	vie 08/06/12
19		Implementación parte 2 del protocolo: Obtención de recursos	9 días	lun 11/06/12	jue 21/06/12
20		Implementación parte 3 del protocolo: Obtención de perfil de contactos	8 días	vie 22/06/12	mar 03/07/12
21		Implementación parte 4 del protocolo: Obtención de recursos de contactos	10 días	mié 04/07/12	mar 17/07/12
22		Creación de BBDD	4 días	mié 18/07/12	lun 23/07/12
23		Implementación de módulo de gestión de BBDD	6 días	mar 24/07/12	mar 31/07/12
24		Implementación de módulo criptográfico	7 días	mié 01/08/12	jue 09/08/12
25		Implementación de módulo de "logs"	3 días	vie 10/08/12	mar 14/08/12
26		**Pruebas**	**3 días**	**mié 15/08/12**	**vie 17/08/12**
27		Pruebas de aceptación	**3 días**	**mié 15/08/12**	**vie 17/08/12**

Ilustración 68: Tareas y tiempos planificados

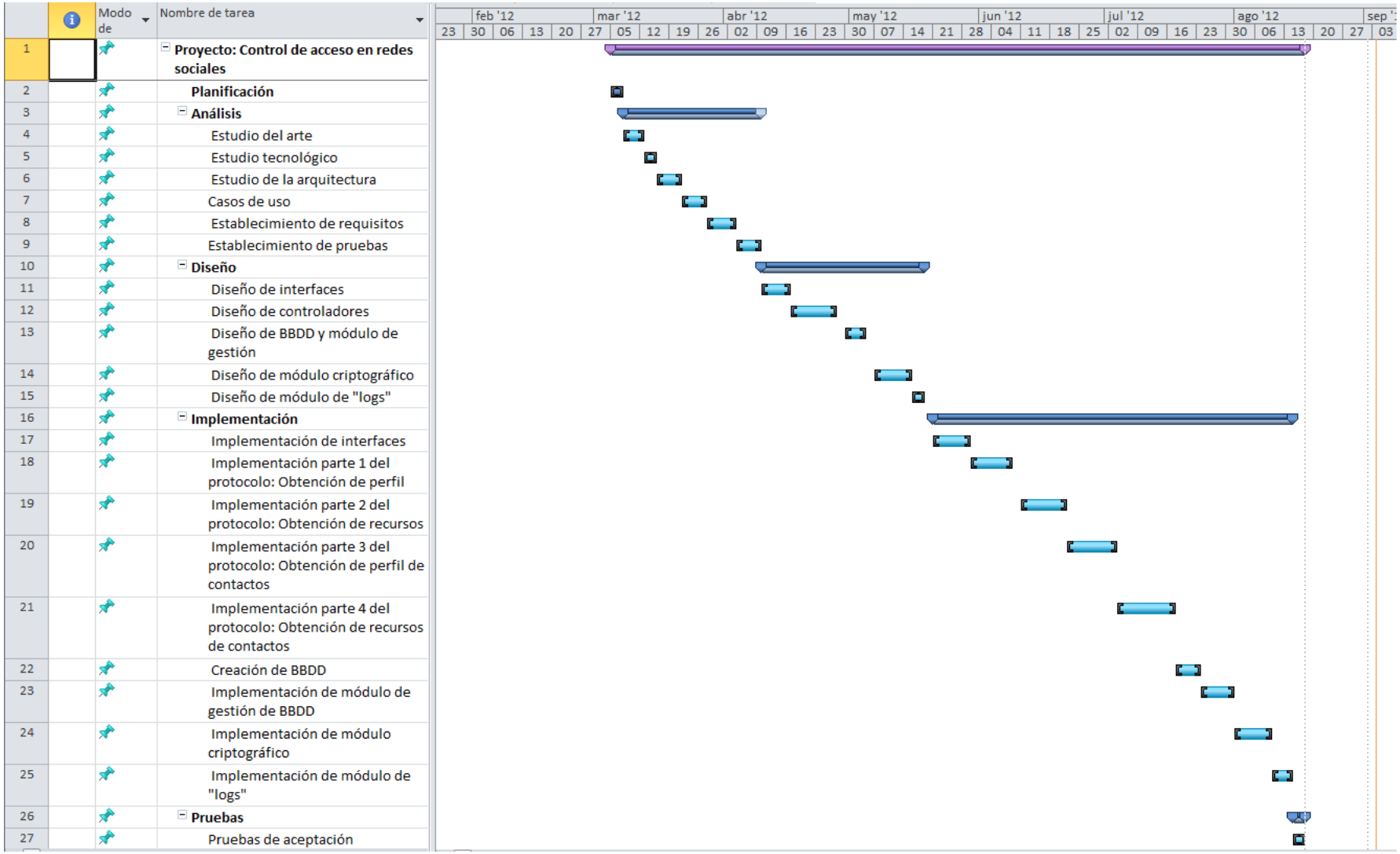

Ilustración 69: Tareas planificadas (Diagrama de Gantt)

Sobre este diagrama se deben tener en cuenta algunas consideraciones:

- Destacar que existen dos grados de tareas. Por un lado, las tareas principales, tales como Planificación, Análisis, Diseño, Implementación y Pruebas, que conforman el esquema principal. Por otro lado, se encuentran subtareas, particulares de este proyecto, que forman cada una de estas tareas principales.
- El período de Pruebas se estima para un corto período de tiempo, concretamente tres días. La selección de este período se debe a que se considera que durante la implementación se comprobará que cada una de las partes del sistema funcionan correctamente, por lo que finalizada esta fase, tan solo se tendrá que realizar las pruebas de aceptación del sistema una vez conjuntado.

A partir de los períodos que aparecen en el diagrama, se puede extraer que la estimación en días para la realización del proyecto es de **120 días** laborables, repartidos en cinco meses y medio (desde inicios de marzo hasta mediados de agosto). Dado que como se ha comentado, la jornada laboral es de seis horas, y tan solo traba en el proyecto un empleado, el esfuerzo total estimado en horas es de **720 horas**.

En la siguiente tabla se puede observar las horas totales dedicadas a cada tarea principal:

Tarea	Horas
Planificación	18 horas
Análisis	138 horas
Diseño	174 horas
Implementación	372 horas
Pruebas	18 horas
TOTAL	**720 horas**

Tabla 26: Recuento de horas planificadas por tarea principal

9.2 Seguimiento real

Una vez finalizado el proyecto, se puede realizar un resumen del coste total, medido en tiempo, de éste. De esta forma se puede realizar una comparación con la planificación inicial establecida, comprobando principalmente si el proyecto ha cumplido sus plazos y, si no lo hiciese, analizar los posibles problemas.
A continuación se muestra, tal y como se hizo en la sección anterior, la tabla de tareas con sus correspondientes tiempos junto al Diagrama de Gantt del seguimiento real del proyecto:

	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		**Proyecto: Control de acceso en redes sociales**	**133 días**	**lun 05/03/12**	**mié 05/09/12**
2		**Planificación**	**3 días**	**lun 05/03/12**	**mié 07/03/12**
3		**Análisis**	**23 días**	**jue 08/03/12**	**lun 09/04/12**
4		Estudio del arte	3 días	jue 08/03/12	lun 12/03/12
5		Estudio tecnológico	3 días	mar 13/03/12	jue 15/03/12
6		Estudio de la arquitectura	4 días	vie 16/03/12	mié 21/03/12
7		Casos de uso	4 días	jue 22/03/12	mar 27/03/12
8		Establecimiento de requisitos	5 días	mié 28/03/12	mar 03/04/12
9		Establecimiento de pruebas	4 días	mié 04/04/12	lun 09/04/12
10		**Diseño**	**29 días**	**mar 10/04/12**	**vie 18/05/12**
11		Diseño de interfaces	5 días	mar 10/04/12	lun 16/04/12
12		Diseño de controladores	9 días	mar 17/04/12	vie 27/04/12
13		Diseño de BBDD y módulo de gestión	5 días	lun 30/04/12	vie 04/05/12
14		Diseño de módulo criptográfico	7 días	lun 07/05/12	mar 15/05/12
15		Diseño de módulo de "logs"	3 días	mié 16/05/12	vie 18/05/12
16		**Implementación**	**74 días**	**lun 21/05/12**	**jue 30/08/12**
17		Implementación de interfaces	8 días	lun 21/05/12	mié 30/05/12
18		Implementación parte 1 del protocolo: Obtención de perfil	9 días	jue 31/05/12	mar 12/06/12
19		Implementación parte 2 del protocolo: Obtención de recursos	10 días	mié 13/06/12	mar 26/06/12
20		Implementación parte 3 del protocolo: Obtención de perfil de contactos	9 días	mié 27/06/12	lun 09/07/12
21		Implementación parte 4 del protocolo: Obtención de recursos de contactos	11 días	mar 10/07/12	mar 24/07/12
22		Creación de BBDD	6 días	mié 25/07/12	mié 01/08/12
23		Implementación de módulo de gestión de BBDD	8 días	jue 02/08/12	lun 13/08/12
24		Implementación de módulo criptográfico	9 días	mar 14/08/12	vie 24/08/12
25		Implementación de módulo de "logs"	4 días	lun 27/08/12	jue 30/08/12
26		**Pruebas**	**3 días**	**vie 31/08/12**	**mié 05/09/12**
27		Pruebas de aceptación	**3 días**	**vie 31/08/12**	**mar 04/09/12**
28		Entrega final	0 días	mié 05/09/12	mié 05/09/12

Ilustración 70: Tareas y tiempos finales

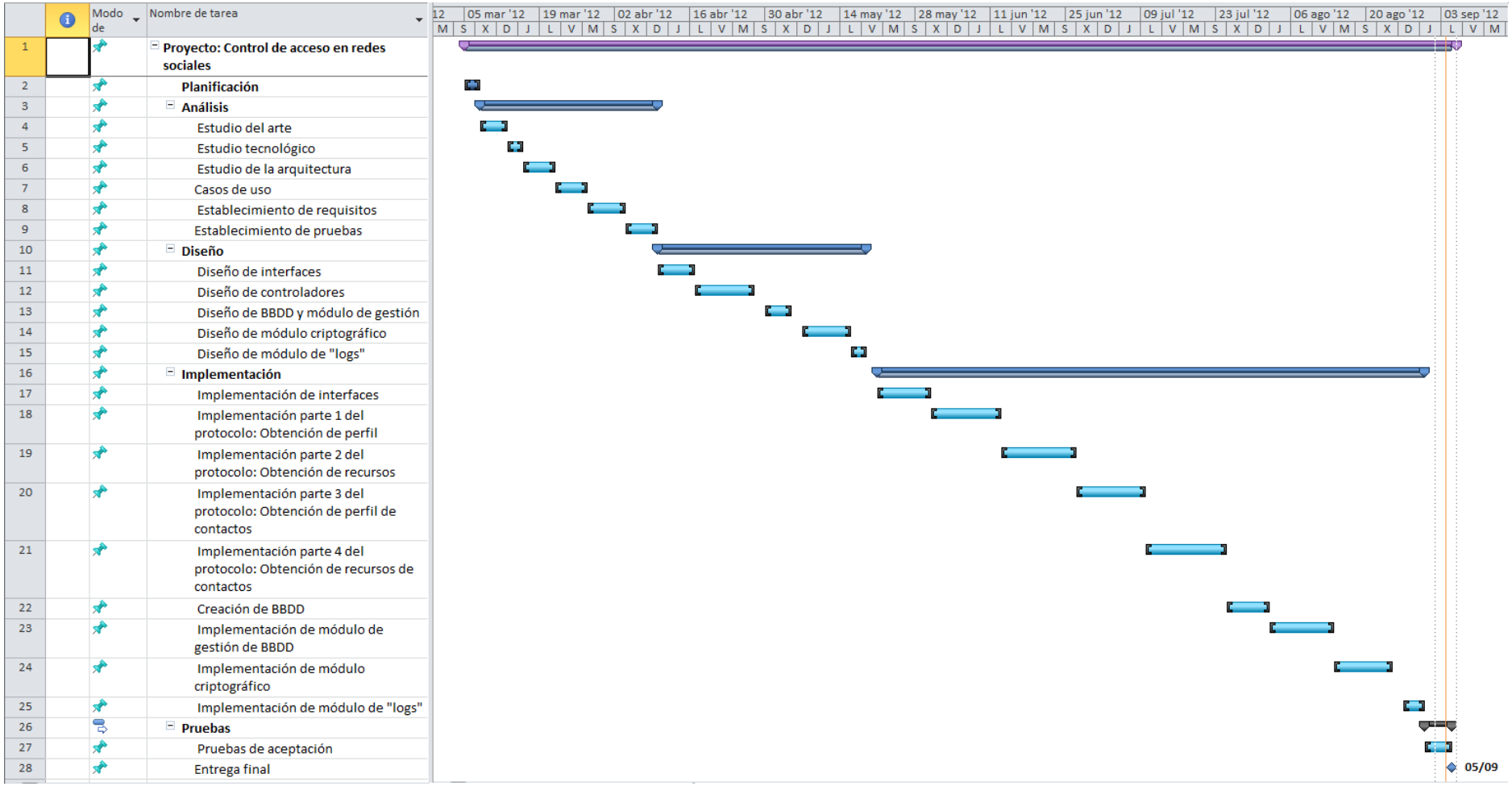

Ilustración 71: Tareas finales (Diagrama de Gantt)

Como se puede observar en el diagrama del seguimiento final, la fase de implementación se ha alargado doce días, lo que ha supuesto un retraso en la fecha de finalización del proyecto, la cual ha quedado fijada finalmente en el día cuatro de septiembre, día previo a la entrega final. Este hecho conlleva un aumento de la cantidad total de días dedicados al proyecto, llegando a los **132 días laborables** repartidos en seis meses (desde inicios de marzo hasta inicios de septiembre). De esta manera, el total de horas dedicadas al proyecto es de **792 horas**.

En la siguiente tabla se puede observar el total de horas destinadas finalmente a cada una de las tareas principales del proyecto:

Tarea	Horas
Planificación	18 horas
Análisis	138 horas
Diseño	174 horas
Implementación	444 horas
Pruebas	18 horas
TOTAL	**792 horas**

Tabla 27: recuento de horas finales por tarea principal

Conociendo las horas estimadas para cada una de las tareas principales, y las horas que finalmente se han dedicado a éstas, se presenta la siguiente tabla en la que se comprobará la desviación en cada una de ellas:

Tarea	Horas estimadas	Horas finales	Desviación
Planificación	18 horas	18 horas	0 horas
Análisis	138 horas	138 horas	0 horas
Diseño	174 horas	174 horas	0 horas
Implementación	372 horas	444 horas	72 horas
Pruebas	18 horas	18 horas	0 horas
TOTAL	**720 horas**	**792 horas**	**72 horas**

Tabla 28: Comparativa entre horas planificadas y finales

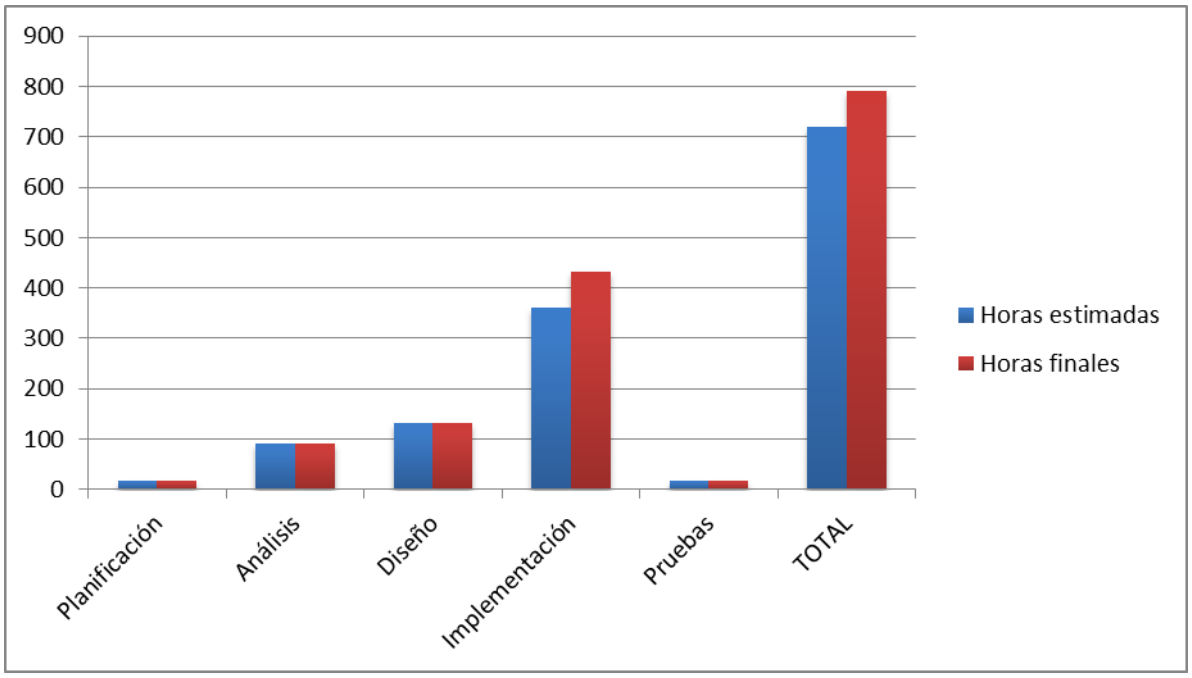

Ilustración 72: Comparativa entre horas planificadas y finales (Gráfico)

Como se puede comprobar, las horas estimadas se han cumplido excepto en el caso de la implementación, en la que existe un desfase de 72 horas. Pese a esta desviación, se ha cumplido el plazo de entrega del proyecto, dado que se ha finalizado antes del cinco de septiembre, por lo que la desviación comentada no ha tenido un gran impacto.

Apéndice B

– Presupuesto –

En este apéndice se va a detallar un presupuesto para el desarrollo del presente proyecto. Este presupuesto indica el coste total del proyecto, en el que se incluyen gastos directos como el salario del personal o la adquisición de tecnologías, y gastos indirectos.

El presupuesto se va a estructurar en distintos apartados, cada uno de ellos dedicados a un tipo de coste concreto, como son el personal, los equipos, el software utilizado, el material fungible y los costes indirectos del proyecto.

Cabe destacar que los costes se van a indicar en euros y las cifras se redondearán a dos decimales. Además, cada uno de los costes detallados a continuación incluirá el IVA, fijado a un 18% sobre el valor.

10.1 Personal

A continuación se presenta el coste que suponen los empleados necesarios para realizar el proyecto, desde su fase de análisis hasta la conclusión de éste.

El coste por personal está formado por el salario bruto total al mes de los empleados, multiplicado por el número total de meses que dure el proyecto.

Para la realización del proyecto tan solo es necesario un empleado, el cual ocupará las distintas categorías necesarias para el proyecto (Jefe de Proyecto, Analista, Programador). Dada esta característica, se determina que el salario bruto que ha de recibir el empleado es el correspondiente a la categoría superior, es decir, Jefe de Proyecto.

Apellidos	Nombre	Categoría por la que imputa	Salario bruto / mes
Marín García	Alberto	Jefe de Proyecto	3.169 , 92 €

Tabla 29: Personal del proyecto

Salario bruto / mes	Dedicación (meses)	Salario bruto total
3.169 , 92 €	5,5	17.434 , 56 €

Tabla 30: Salario bruto total

Junto a este coste, se ha de añadir el que supone los pagos a la seguridad social por parte de la empresa. Este coste se calcula multiplicando la base de cotización por el tipo de cotización.

Este cálculo se muestra en la siguiente tabla. Destacar que la base y el tipo de cotización se extraen del Régimen General de la Seguridad Social (16), en función de la categoría profesional del empleado (“Ingenieros y Licenciados. Personal de alta dirección no incluido en el artículo 1.2 c) del Estatuto de Trabajadores”).

Base máxima (euros / mes)	Tipo de cotización	Coste mensual S.S.
3.262 , 5 €	23 , 60 %	769 , 95 €

Tabla 31: Coste mensual Seguridad Social

Coste mensual S.S.	Dedicación (meses)	Coste total S.S.
769 , 95 €	5,5	4.234 , 73 €

Tabla 32: Coste total Seguridad Social

Con los costes del salario y la Seguridad Social calculados individualmente, se puede obtener los costes totales del personal para el proyecto.

Salario bruto total	Coste S.S. total	Coste total del personal

17.434 , 56 €	4.234 , 73 €	21.669 , 29 €

Tabla 33: Coste total del personal

10.2 Equipos

En esta sección se procede a calcular el coste que suponen los equipos necesarios para llevar a cabo el proyecto. Cabe destacar que no se imputa el coste total de estos equipos, sino el correspondiente a los meses que han sido utilizados en función de su período de amortización.

Nombre	Características	Coste total	Dedicación (meses)	Período de amortización (meses)	Coste imputable
Ordenador sobremesa	Procesador quad-core AMD Phenom II, 2x2GB RAM Kingston, HDD 1TB, Gráfica ATI Radeon 5770 HD Series 1024MB dedicados	700 €	5,5	48	80 , 21 €
Ordenador portátil	MacBook. Intel core duo, 2 GB RAM, HDD 400 GB	758 €	5,5	48	86 , 85 €
Monitor	BENQ T2210HD	144 €	5,5	36	22 €
Ratón y Teclado	Microsoft Wireless Laser Desktop 3000	45 €	5,5	12	20 , 63 €
PLC (red)	TP_LINK 200 Mbps	25 €	5,5	18	7 , 64 €
Router CISCO + WiFi NetGear		50 €	5,5	24	11 , 46 €
TOTAL					**228 , 79 €**

Tabla 34: Equipos

10.3 Software

En este apartado se calcula el coste que supone la compra de productos software o bien la adquisición de sus licencias. Del mismo modo que se ha realizado en la sección anterior, el coste imputado se obtendrá en función del período de desamortización de estas licencias.

El software que se va a utilizar para la realización del proyecto es el siguiente:

- **Microsoft Windows 7 Ultimate (17):** este software es el elegido como sistema operativo sobre el que se realizará el proyecto, tanto las fases de análisis y diseño como el desarrollo.

- **NetBeans IDE 7.1 (18):** entorno de desarrollo usado para implementar el proyecto. Se trata de un software libre, por lo que no supone ningún coste al proyecto.
- **GlassFish v3 (19):** servidor de aplicaciones sobre el que se ejecutará el proyecto durante su implementación y su posterior funcionamiento. No supone ningún coste al proyecto.
- **Microsoft Office 2010 – Home Edition (20):** software de ofimática usado principalmente para documentar el proyecto y su posterior presentación. De las herramientas que ofrece, se hará uso de Word, Excel y PowerPoint.
- **Microsoft Project 2010 – Proffesional Edition (21):** software utilizado para la planificación del proyecto así como su posterior seguimiento.
- **Software Ideas Modeler (22):** utilizado durante la fase de análisis y diseño, permite la creación de todo tipo de diagramas. Al tratarse de software libre, no supone ningún coste al proyecto.

Nombre	Coste total	Dedicación (meses)	Período de amortización (meses)	Coste imputable
Microsoft Windows 7 Ultimate	254.39 €	5,5	24	58 , 30 €
NetBeans IDE 7.1	0 €	5,5	N/A	0 €
GlassFish v3	0 €	5,5	N/A	0 €
Microsoft Office 2010 – Home Edition	101.38 €	5,5	36	15 , 48 €
Microsoft Project 2010 – Professional Edition	795.5 €	5,5	36	121 , 53 €
Software Ideas Modeler	0 €	5,5	N/A	0 €
TOTAL				**195 , 31 €**

Tabla 35: Software

10.4 Material fungible

Sección dedicada al cálculo del coste del material fungible, formado por el material de oficina necesario durante la realización del proyecto. Este material de oficina incluye artículos variados como folios, bolígrafos, rotuladores…

Artículo	Coste
Material de oficina	60 €
TOTAL	**60 €**

Tabla 36: Material fungible

10.5 Viajes y dietas

Dado que durante todo el proceso de desarrollo del software se realizan varias reuniones con el cliente, en ésta sección se va a especificar el coste total de los viajes y dietas ocasionados por dichas reuniones.

Coste por reunión	Nº de reuniones	Coste total
25 €	6	150 €

Tabla 37: Viajes y dietas

10.6 Costes indirectos

Como últimos costes añadidos aparecen los indirectos, en los que se engloban costes como el gasto de luz de los sistemas o el acceso a internet.

Concepto	Coste mensual	Meses	Coste total
Consumo luz	45 €	5,5	247 , 50 €
Conexión a internet (50 MB) ONO	39 , 90 €	5,5	219 , 45 €
TOTAL			**466 , 95 €**

Tabla 38: Costes indirectos

Cabe destacar que el local donde se realiza el desarrollo del proyecto no va a suponer ningún coste añadido al cliente por pertenecer a la empresa desarrolladora.

10.7 Costes totales

Con todos los costes calculados de manera individual, se puede obtener el coste total del proyecto. A este coste, se le debe añadir un margen de beneficios, así como uno dedicado a posibles pérdidas que puedan ocasionarse. Por último, se debe añadir el IVA sobre el precio final del proyecto.

Coste total:

Concepto	Coste (sin IVA)	Coste (IVA)
Personal	18.363 , 80 €	21.669 , 29 €
Equipos	193 , 89 €	228 , 79 €
Software	165 , 52 €	195 , 31 €
Material fungible	50 , 85 €	60 €
Viajes y dietas	127 , 12 €	150 €
Costes directos	**18.901 , 18 €**	**22.303 , 39 €**
Costes indirectos	**395 , 72 €**	**466 , 95 €**
TOTAL	**19.296 , 90 €**	**22.770 , 34 €**

Tabla 39: Coste total

Margen de beneficio:

Se determina que el margen de beneficio que se desea para este proyecto es de un 30% del coste total de su desarrollo.

Coste total (sin IVA)	Coste total (IVA)	Porcentaje de beneficio	Beneficio (sin IVA)	Beneficio total (IVA)
19.296 , 90 €	22.770 , 34 €	30 %	5.789 , 07 €	6831 , 10 €

Tabla 40: Margen de beneficio

Margen de riesgo:

El margen de riesgo determinado para este proyecto es de un 10% sobre su coste total.

Este porcentaje es suficiente para costear las posibles pérdidas que se puedan originar durante el desarrollo del proyecto, si bien no es muy elevado dado que no existen grandes riesgos.

Coste total (sin IVA)	Coste total (IVA)	Porcentaje de riesgo	Margen de riesgo total (sin IVA)	Margen de riesgo total (IVA)
19.296 , 90 €	22.770 , 34 €	10 %	1.929 , 69 €	2.277 , 03 €

Tabla 41: Margen de riesgo

Coste final:

Concepto	Coste (sin IVA)	Coste (IVA)
Coste del proyecto	19.296 , 90 €	22.770 , 34 €
Margen de beneficio	5.789 , 07 €	6831 , 10 €
Margen de riesgo	1.929 , 69 €	2.277 , 03 €
TOTAL	**27.015 , 65 €**	**31.878 , 47 €**

Tabla 42: Coste final

En resumen, el coste final del proyecto y, por tanto, cantidad a abonar por el cliente es de un total de **33.016 , 99 €.**

10.8 Desviaciones del presupuesto

Finalizado el proyecto y comprobado el seguimiento de éste, se comprueba que existe una desviación de medio mes respecto a la planificación inicial, lo cual supone variaciones en los costes respecto al presupuesto establecido. A continuación se detallan estas variaciones, para calcular el beneficio final del proyecto.

Concepto	Coste previsto	Coste final	Desviación
Personal	21.669 , 29 €	23.639 , 22 €	1.969 , 93 €
Equipos	228 , 79 €	249 , 58 €	20 , 79 €
Software	195 , 31 €	213 , 08 €	17 , 77 €
Material fungible	60 €	60 €	0 €
Viajes y dietas	150 €	150 €	0 €
Costes directos	**22.303 , 39 €**	**24.311 , 88 €**	**2008 , 49 €**
Costes indirectos	**466 , 95 €**	**509 , 40 €**	**42 , 45 €**
TOTAL	**22.770 , 34 €**	**24.821 , 28 €**	**2.050 , 94 €**

Tabla 43: Desviación del presupuesto

Como se observa en la tabla, la desviación total sobre el presupuesto calculado es de **2.050,94 €**. Dado que se estableció un margen de riesgo de 2.277,03 €, la desviación entra dentro de este margen, por lo que no se han producido pérdidas para el proyecto.

De esta forma, el beneficio total para la empresa desarrolladora se compone del margen de beneficio unido al margen de riesgo una vez restada la desviación de costes (226,09 €). Por tanto, el beneficio total del proyecto es de **7.057 , 19 €.**